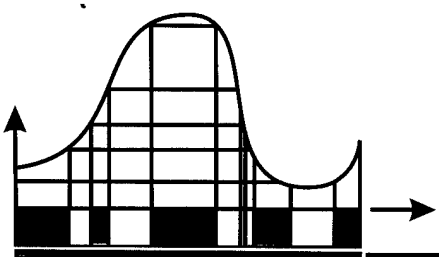

Center for Foundations of Intelligent Systems

Technical Report
98-08

QoS based Evaluation of the Berkeley Continuous Media Toolkit

D. Wijesekera, S. Varadarajan, S. Parikh, J. Srivastava and A. Nerode

July 1998

CORNELL
UNIVERSITY

625 Rhodes Hall, Ithaca, NY 14853 (607) 255-8005

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 March 1999	TECHNICAL

4. TITLE AND SUBTITLE
QoS BASED EVALUATION OF THE BERKELEY CONTINUOUS MEDIA TOOLKIT

5. FUNDING NUMBERS
DAAH04-96-1-0341

6. AUTHOR(S)
D. WIJESEKERA, S. VARADARAJAN, S. PARIKH, J. SRIVASTAVA, and A. NERODE

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Regents of the University of California
c/o Sponsored Projects Office
336 Sproul Hall
Berkeley, CA 94720-5940

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 35873.137-MA-MUR

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This paper presents a performance analysis of the Continuous Media service provided by the Berkeley, *Continuous Media Toolkit (CMT)*. We show a development of metrics, validation by means of a user study, and a performance evaluation of a prototyping environment. The *CMT* is a popular environment that satisfies our need of being easily extendible and offers a quick prototyping environment. From a human user's perspective, in order for multimedia demonstrations to be comprehensible, the number of audio or video frames dropped and the timing delays in the ones that are displayed, and the degree of asynchrony between two streams such as audio and video, need to be kept within limits. Therefore, it is important to know the frame dropping, timing delay and synchronization drift characteristics of CMT. In a series of experiments we monitored the variation of these parameters with respect to processor, network and event loop loads. It was observed that loads affect aggregate frame drops at lower rates , and consecutive frame drops at higher rates. Because at a higher rates a large number of consecutive frames are dropped, the ones that are played appear in a more timely manner. It is shown that according to metrics provided by other studies, CMT provides imperceptible audio-video mis-synchronization for about 10 seconds, and tolerable synchronization for about 13 seconds, from the start of the clips for local clients under low processor loads. It is also shown that under high loads, synchronization is achieved at the cost of losing media frames...

14. SUBJECT TERMS	15. NUMBER OF PAGES
quality of service, metrics, CMT, toolkit, synchronization losses, media losses	28
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

QoS Based Evaluation of the Berkeley Continuous Media Toolkit (CMT)

Duminda Wijesekara[†], Srivatsan Varadarajan, Shwetal Parikh,
Jaideep Srivastava and Anil Nerode[‡].

[†]Senior Systems Engineer, Space Systems, Honeywell Inc, Clearwater, FL.
Department of Computer Science and Engineering, University of Minnesota Minneapolis, MN.
[‡]Goldwin Smith Chair in Mathematics and Computer Science, Department of Mathematics, Cornell University, Ithaca, NY.
Email: wijesekera@space.honeywell.com, {varadara, ssparikh, srivasta}@cs.umn.edu , anil@math.cornell.edu.

Abstract. This paper presents a performance analysis of the Continuous Media service provided by the Berkeley, *Continuous Media Toolkit (CMT)*. We show a development of metrics, validation by means of a user study, and a performance evaluation of a prototyping environment. The *CMT* is a popular environment that satisfies our need of being easily extendible and offers a quick prototyping environment. From a human user's perspective, in order for multimedia demonstrations to be comprehensible, the number of audio or video frames dropped and the timing delays in the ones that are displayed, and the degree of asynchrony between two streams such as audio and video, need to be kept within limits. Therefore, it is important to know the frame dropping, timing delay and synchronization drift characteristics of CMT. In a series of experiments we monitored the variation of these parameters with respect to processor, network and event loop loads. It was observed that loads affect aggregate frame drops at lower rates , and consecutive frame drops at higher rates. Because at a higher rates a large number of consecutive frames are dropped, the ones that are played appear in a more timely manner. It is shown that according to metrics provided by other studies, CMT provides imperceptible audio-video mis-synchronization for about 10 seconds, and tolerable synchronization for about 13 seconds, from the start of the clips for local clients under low processor loads. It is also shown that under high loads, synchronization is achieved at the cost of losing media frames. One interesting point to be noted is that a *pseudo resource* such as event loop is as critical as processor cycles/ network bandwidth as seen from the series of experiments we conducted. Thus any solution proposed must also carefully address the problems that arise due to overheads/loads in the event loop. As part of ongoing effort we present some solutions to the observed problems.

Keywords: Quality of Service, Metrics, CMT, Toolkit, Synchronization Losses, Media Losses

1 Introduction

Rapid growth of multimedia systems, and accordingly research efforts in this area, have made it necessary to have toolkit support for rapid prototyping. Being one of the most popular toolkits offered in multimedia, the *Berkeley Continuous Media Toolkit (CMT)* [SRY93, MPR97] has gone a long way in fulfilling this need. In [MPR97], Mayer-Patel and Rowe measured the performance of CMT and its system overheads. In this paper, we measure the quality of audio-video service (drops, delays and synchronization drifts) obtainable in CMT.

The addition of of *continuous media* to human interfaces of computing machines leads to better representations and consequently quicker comprehension of the communicated message. Consequently, such media should be presented in a way that ranges from being appealing to marginally satisfactory for human consumers. This wide margin has been quantitatively captured by *Quality of Service (QoS)* metrics at the application level [AFKN94, AFKN95, SE93, Ste96, WS96]. Between mostly qualitatively stated application requirements and a multimedia testbed, there are many stages of design, testing, performance evaluation and implementation. Also, most multimedia systems display synchronized audio and video. In order to give a *natural* effect, and therefore to

This work is supported by Air Force contract number F30602-96-C-0130 to Honeywell Inc, via subcontract number B09030541/AF to the University of Minnesota, and DOD MURI grant DAAH04-96-10341 to Cornell University

provide a sense of cohesiveness and ease of comprehension, audio and video have to be synchronized within human perceptual limits [Ste96, SE93]. Hence, there needs to be a systematic evaluation of computer systems to measure their ability to deliver Continuous Media. This paper describes an evaluation process that has been developed in our research at the University of Minnesota and its use in evaluating the *Berkeley Continuous Media Toolkit (CMT)*

CMT provides the *Logical Time System (LTS)*, a mechanism for ensuring the timely progress of streams, and for providing inter-stream synchronization [MPR97]. In a series of experiments we investigate the quality of audio-video synchronization provided by the LTS mechanism against known metrics.

Since our evaluations measure perceptual quality, they should be measured external to the system that is providing the continuous media (CM) services [SNL95]. As shown in [SNL95], internal and external measurements of metrics could be different. As a first step in this direction, we report internal measurements, since the quality of the latter can be no better than the former, and consequently we can obtain an upper bound on external quality (in terms of end user perception).

In our usage, we have noticed that CMT drops frames in its stream services. Therefore, we have chosen two metrics to measure audio-video synchronization. The first one [Ste96] is designed to measure synchronization between loss-less media streams. The second [WS96] has been designed to measure synchronization quality in lossy streams. Additional set of metrics characterizing continuity and timing drifts [WS96] are used, and these metrics are used to characterize CMT stream services.

Our experiments show that for a single site display, CMT provides tolerable audio-video synchronization for about 13 seconds under low processor load, and that this value decreases under higher loads. We have also found that in distributed playing, there is hardly any tolerable synchronization, and that available synchronization decreases as processor or network load increases, calling for additional mechanisms to maintain tolerable audio-video synchronization. The general trend of our performance evaluation indicates that as the speed increases, the frame drops increase. Higher loads induce aggregate drops at lower speeds and consecutive drops at higher speeds. Furthermore, when a large number of frames are dropped by CMT, the remaining are displayed in a timely manner.

The rest of the paper is organized as follows. Section 2, describes the metrics used to measure continuity, drifts and synchronization in lossy media. Section 3 is a brief introduction to the structure and functionality of CMT. Section 4 describes our experimental setup . Section 5 gives their analysis with respect to Media Loss metrics described in [WS96]. and Section 6 gives the analysis of our experimental outcome with respect to Synchronization metrics described in [Ste96, SE93, WS96]. Finally, Section 7 ends the paper with concluding remarks.

2 QoS Metrics for Continuous Media

For the purpose of describing QoS metrics for lossy media streams, we envision a CM stream as a flow of data units (referred to as logical data units - LDUs in the uniform framework of [SB96]). In our case, we take a video LDU to be a frame, and an audio LDU to constitute 8000/30, i.e. 266 samples of audio[1]. Given a rate for streams consisting of these LDUs, we envision that there is time slot for each LDU to be played out. In the ideal case a LDU should appear at the beginning of its time slot.

[1] SunAudio has 8-bit samples at 8kHz, and an audio frame constitutes of 266 such samples equivalent to a play time of one video frame, i.e. 1/30 seconds.

Metrics for Continuity Continuity of a CM stream is measured by three components: *rate, drift* and *content.* The ideal rate of flow and the maximum permissible deviation from it constitute our *rate* parameters. As stated, given the ideal rate and the beginning time of a CM stream, there is an ideal time for a given LDU to arrive/ be displayed. Given the envisioned fluid-like nature of CM streams, the appearance time of a given LDU may deviate from this ideal. Our *drift* parameters specify aggregate and consecutive non-zero drifts from these ideals, over a given number of consecutive LDUs in a stream. For eg., first four LDUs of two example streams with their expected and actual times of appearance, are shown in Fig. 1. In the first example stream, the drifts are respectively 0.0, 0.2, 0.2 and 0.2 seconds; and accordingly it has an aggregate drift of 0.6 seconds per 4 time slots, and a non-zero consecutive drift of 0.6 seconds. In the second example stream, the largest consecutive non-zero drift is 0.3 seconds and the aggregate drift is 0.5 seconds per 4 time slots. The reason for a lower consecutive drift in stream 2 is that the unit drifts in it are more spread out than those in stream 1.

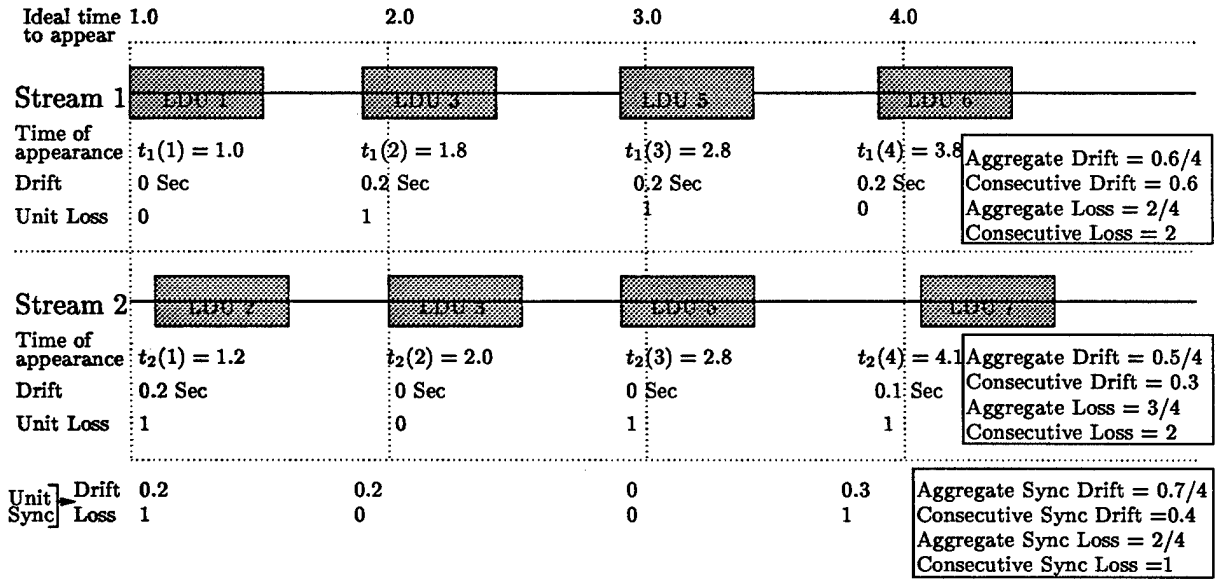

Fig. 1 Two Example Streams used to Explain Metrics

In addition to timing and rate, ideal contents of a CM stream are specified by the ideal contents of each LDU. Due to loss, delivery or resource over-load problems, appearance of LDUs may deviate from this ideal, and consequently lead to discontinuity. Our metrics of continuity are designed to measure the average and bursty deviation from the ideal specification. A loss or repetition of a LDU is considered a unit loss in a CM stream. (A more precise definition is given in [WS96].) The aggregate number of such unit losses is the *aggregate loss* of a CM stream, while the largest consecutive non-zero loss is its *consecutive loss.* In the example streams of Fig. 1, stream 1 has an aggregate loss of 2/4 and a consecutive loss of 2, while stream 2 has an aggregate loss of 2/4 and a consecutive loss of 1. Once again, the reason for the lower consecutive loss in stream 2 is that its losses are more spread-out than those of stream 1.

Metrics for Synchronization in Lossy Media As in the case of continuity metrics, synchronization metrics are also categorized into content, rate and drift. Rate of rendition of a collection of

synchronized streams is determined by the rates of component streams. (must be equal to be synchronized) The rate variation of a collection of synchronized streams is taken to be the maximum of their component streams.

In a perfectly synchronized collection of streams, the i^{th} LDU of each stream should start playing out at the same instant of time. Failure to accomplish this ideal is measured by the maximum difference between the display start time of the LDUs in the group, and is referred to as the unit synchronization drift. The aggregate of such unit synchronization drifts over a given number of LDU slots is the aggregate synchronization drift, and the maximum of such non-zero consecutive synchronization drifts is the consecutive synchronization drift. They measure the average and bursty time drifts in synchronization. In Fig. 1, the two streams have unit synchronization drifts of 0.2, 0.2, 0.0, and 0.3 seconds respectively, resulting in an aggregate synchronization drift of 0.7/4, and a consecutive synchronization drift of 0.4 seconds.

For the content component, with streams consisting of LDUs with equal play-out times, there is a natural collection of LDUs that are to be played out simultaneously. The largest discrepancy in the LDU numbers between any two pairs in such a collection is referred to as the unit synchronization loss. The aggregate and largest non-zero consecutive unit synchronization loss is referred to as *aggregate synchronization content loss* and *consecutive synchronization content loss*, respectively. In the example of Fig. 1, due to losses of LDUs there are unit synchronization content losses at the first and the last pairs of LDU's, resulting in an aggregate synchronization content loss of 2/4 and a consecutive synchronization loss of 1.

Parameters for Lossy Metrics In a user study [WSNF99] it has been determined that aggregate losses of up to 17/100 were imperceptible, and those beyond 23/100 were annoying, and in between these two values were tolerable. The tolerable value for consecutive losses were determined to be two frames. For audio this limit was about three frames. Due to the inability of precisely introducing drifts, the user study did not estimate any values for drifts.

For audio-video synchronization, 6/100 was determined to be the limit of imperceptible aggregate loss and 7/100 was determined to be the limit for tolerable aggregate loss. A unit consecutive loss (i.e. 1/100) was determined to be the tolerable limit.

3 CMT: Design and Functionality

The Berkeley Continuous Media Toolkit (CMT), has been constructed for the purposes of application development and research in multimedia systems[SRY93]. It is an extensible system consisting of a three layered architecture. The topmost layer, referred to as the *application code layer*, consists of Tcl, Tk and Tcl-Dp [SRS93] code. The next layer, consists of a CM object model, time and synchronization services, CM event services and storage/buffer services for CM streams. The bottom most *resource layer* consists of operating system and hardware services. The complete design of CMT is explained in detail in [SRY93].

One of the outstanding features of CMT is the ease of programming in Tcl/Tk [Ous94, Wel95], and the extensible and relatively *policy free* nature of implementation, which makes it a great testbed for experimentation.

CMT envisions multimedia as streams flowing in and out of CM objects, in their journey between *sources* and *sinks*. These CM objects correspond to *services* that are used by CM streams, such as *segments*, that represent data sources stored in files. Similarly, *play* objects represent stream players, such as speakers and video displays. CMT provides distributed multimedia services in the sense that objects in a CM pipeline can sit on different locations, thereby requiring network transport

services. The network twin objects *packet source (pktSrc)* and *packet destination (pktDest)* provide these services of sending and receiving streamed data.

A CMT application program specifies pipelines of CM streams flowing between CM objects in Tcl/Tk. The objects can be created within some *CM process*. These scripts are interpreted by a Tcl interpreter extended with Tcl-DP to provide distributed services. For an application that uses only one location, a single CM process is created with appropriate sources and sinks. Distributed applications have CM processes at each location, where objects belonging to that location are created within the corresponding process.

The Logical Time System (LTS) in the CMT Library layer provides a mechanism for applications to maintain a concept of where in time the application is, and how fast time is progressing. More than one LTS can exist in one application and different CM processes can be synchronized by using the same LTS. CMT ensures that the LTS progresses according to the *rate* specified by the application program.

CMT maintains CM streams by passing data between objects by either the push model or the pull model. In our experimental setup, we used the push model, where the producer of data, typically the source object, initiates the data transfer to the object immediately downstream from it. The source object maintains a continuous streams of CM data by periodically invoking itself to fetch data from the specified file at the specified rate, and transfers data to the corresponding object downstream. The period of such data fetches are specifiable in application code. Objects that are being called by some other object upstream provides an *accept* method, that is being passed a buffer containing appropriate data by the callee. After *processing* the data in the buffer, such an intermediate object calls the accept method of its immediate downstream neighbor, thereby maintaining the flow of CM data through the specified pipeline, until a sink object such as a display device is reached. The sink objects are the final consumers of CM data, whereby it is displayed to the human viewer.

An important point to be noted is that CMT uses Tcl/Tk in the application code (Tcl/Tk Scripts) layer. CMT's library/interface provides event handling through a collection of different services and mechanisms and in particular allows objects to schedule a C callback to be made at a particular point in system time [MPR97]. Also it is possible for an application to associate an action to be taken whenever an event, which have been registered a-priori, occurs. An event loop is built into Tcl. Tcl checks for events and calls out to handlers that have been registered for different types of events. You can register Tcl commands or C APIs to be called in response to events. Thus the Tcl/Tk event loop , not only handles different Tcl event (like window events, timer events etc), but also handles all the different events registered and serviced in CMT. Hence the event loop is an intrinsic resource that is being constantly utilized for a CM service provided by CMT.

3.1 Application Code Used in Our Experiment

In our experiments, we use two application programs, referred to as *local playback application* and *remote playback application* in [SRY93], graphically represented in Fig 2. The local experiment consists of a single CM process containing four objects, namely: *mjpegSegment*, *auSegment*, *mjpegPlay* and *auPlay*. As shown in Fig 2, they are pipelined so that *mjpegSegment* passes a video stream to *mjpegPlay* to be displayed at the local screen and *auSegment* passes an audio stream to *auPlay* to be displayed at the speaker. Although the application program does not explicitly use it, the play objects call an accept method of a low level object called *device*, that is responsible for the final display of media units.

In the *remote play* experiment, the segments and play objects are on two different machines that are connected by Ethernet. Consequently, in addition to the objects used in the local play,

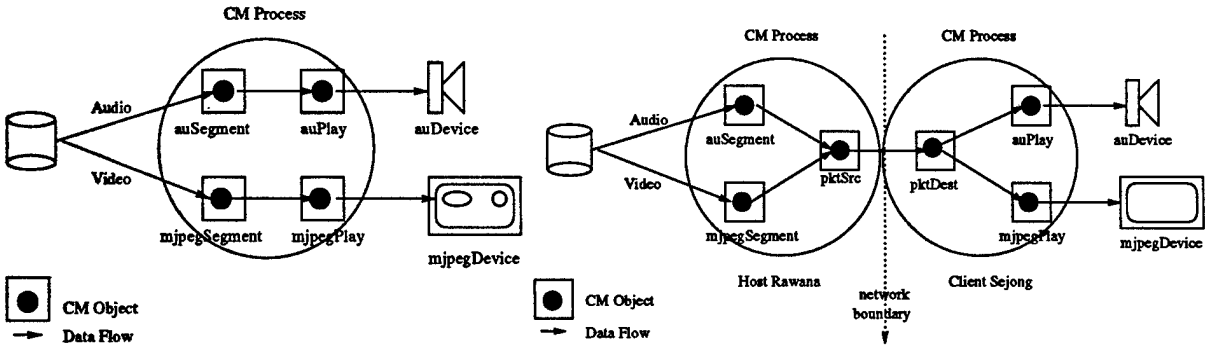

Fig. 2 Local and Remote Playback Experiments in CMT

packet source and *packet destination* network twins are used.

Because the performance analysis entails the behavior of objects used in our application code, following sections are devoted to a detailed description of these objects.

Segment Object Given the current value of the LTS object, the segment object determines the appropriate video frames or parts of audio frames to be fetched and played in the current fetch cycle. It then schedules itself to be called back after a lapse of the *cycle time*. The application code also can set a parameter *send ahead*, that specifies the time that the server is supposed to be ahead of the client.

In the process of being recalled, based on the time it was actually supposed to be called and the time it was called, the video segment determines a number of frames to be dropped by using an *inverse binary order* [Smi]. This scheme attempts to minimize the consecutive dropping of frames, so that at the end, the stream would not appear *jittery*. In dropping the frames, the *mjpegSegment* also interpolates the display times of frames adjoining the dropped frame so that, although the frames were dropped, on the time scale no time gaps appear in the stream.

Play Object In the application code we used, the play object checks and discards frames that are too late to be displayed, and then displays them in order. The video play object provides hooks to select the appropriate frame out of a collection that has been handed over to itself in a buffer.

Device Object These objects are transparent to the programmer and contain low level calls specific to the platform and they submit data to be played, as has been called by the play object. The *SparcAuDevice* pre-empts *leftover* audio to *re-synchronize* at a regular frequency.

Network Twins The network twins, *packet-Source (pktSrc)* and *packet-destination (pktDest)* uses a *cyclic UDP* protocol, where it is responsible for its own retransmission of the lost/delayed packets. This protocol has been explained in detail, with experimental verification of appropriateness for CM transport on best effort networks, in [Smi].

4 Experimental Design

The objective of our experiments is to measure the performance of CMT with respect to audio-video synchronization and continuity losses (timing and drops) in audio and video. In our performance evaluation, we do so with respect to the metrics described in Section 2.

4.1 Types of Experiments

We carried out two main experiments where, in the first one the segment object and the play object resided on the same machine, which we refer to as *local* objects. In the second experiment they were distributed across two machines. By the very design of CMT, this setup requires having *network twins*, the *packet source* and the *packet destination*. In the nomenclature of [SRY93, MPR97], they are the *local playback application model* and the *simple remote playback application model*, of which the flow of streams are given in Fig. 2.

It is imperative that resource scarcity is the main reason for the perceived *low quality*. It is also obvious that the different resources which may be cause the bottlenecks are processor cycles and network bandwidth. Since Unix and NT does not support periodic scheduling as in a Real Time (RT) system , event loops act as a periodic scheduler not supported by the Operating Systems (OS) scheduler, built at the application layer. Nevertheless, event loop now becomes a *pseudo resource* (as in RT schedulers) and affects the performance of CMT to a great extent. By the very nature of continuous media it would be helpful to have such a an event loop, which aids periodic scheduling, support at the operating systems layer. This reduces the overheads incurred but necessitates some form of admission control for registering of events in the event loop.

In our experiments we measured the performance of CMT with respect to selected metrics by varying processor load , network load and event loop load. Accordingly, for local experiments we collected data under different processor and event loop loads, and for remote experiments we collected data under different network , processor and event loop loads. In repeating experiments, the processor loads were measured by `top`, which gives the number of processes waiting for the processor, i.e. the length of the waiting queue. The experiments were carried out only when the number reported for the processor queue had stabilized for the immediately preceding 1, 5 and 15 minutes. The numbers reported are these stable values. The network loads were measured by the use of a *network sniffer*, and the experiments were carried out under stable loads reported by the sniffer in terms of packets per second.

In addition to the processor and network being critical (and real) resources for the behavior of Continuous Media in CMT, one other resource (pseudo resource) which may implicitly affect the behavior of display of frames in CMT is the TCL Event Loop. This is the fallout from the fact that CMT's timing behavior and the LTS mechanisms are based solely on a single TCL Event Loop. We also need to determine whether having multiple independent event loops would improve continuity/timing behavior significantly. We loaded the Tcl event loop externally by making CMT scripts execute some TCL event (other than CMT) periodically. The frequency of execution of these scripts was chosen to be once every 30 milliseconds, as this is the frequency at which the Tcl resource is invoked (i.e. whenever the next frame needs to be grabbed). The experiments were carried out for varying periods/duration of execution (time taken for the execution of the script/alternate event), characterized as varying event loop loads. The different event loop loads were 0 (No Load),20,40,60,80,100 milliseconds. Readings were noted for stable values and some initial readings were neglected.

4.2 Experimental Parameters and Methodology

In all of our experiments, we displayed audio and video streams consisting of 1600 frames in each stream. We ran the local experiment with different processor loads of 0, 1 and 5, which we call *low*, *medium* and *high* loads. For network loads we ran the experiment with approximately 0, 200 and 2000 packets/second. For each load setting we repeated our experiment five times and collected average statistics over them.

Our experiments were carried out using two uni-processor Ultra Sparc 1's with 64 MB RAM each, connected by a 10 Mbps Ethernet. Our work stations were equipped with JPEG Parallax© cards, and we used the Motion-JPEG video format and the 8-bit SparcAu format for audio.

As stated in [MPR97], some care must be taken to ensure that the experimental methodology does not degrade the performance of the system, i.e. is as *non-intrusive* as possible. In this respect, we ensured that the time recorded by each module has been taken from already existing variables, and written into global arrays that were allocated before the system started to run in a stable state. Furthermore, these arrays were written into files when the objects were destroyed, thereby incurring minimal overhead in maintaining performance statistics. We also eliminated some initial readings to ensure that any start-up overhead does not affect final measurements. Finally, in order to keep track of frame numbers, we have introduced frame numbers into header portions of video and audio frames. Statistics maintained were the frame number and the time of arrival at each relevant CM object.

5 Media Loss (Drifts and Drops) Experiment

As stated, the objective of our experiments is to measure the performance of CMT with respect to continuity losses in audio and video. Our performance evaluation, is carried out with respect to metrics set forth in Section 2.

5.1 The Local Experiment

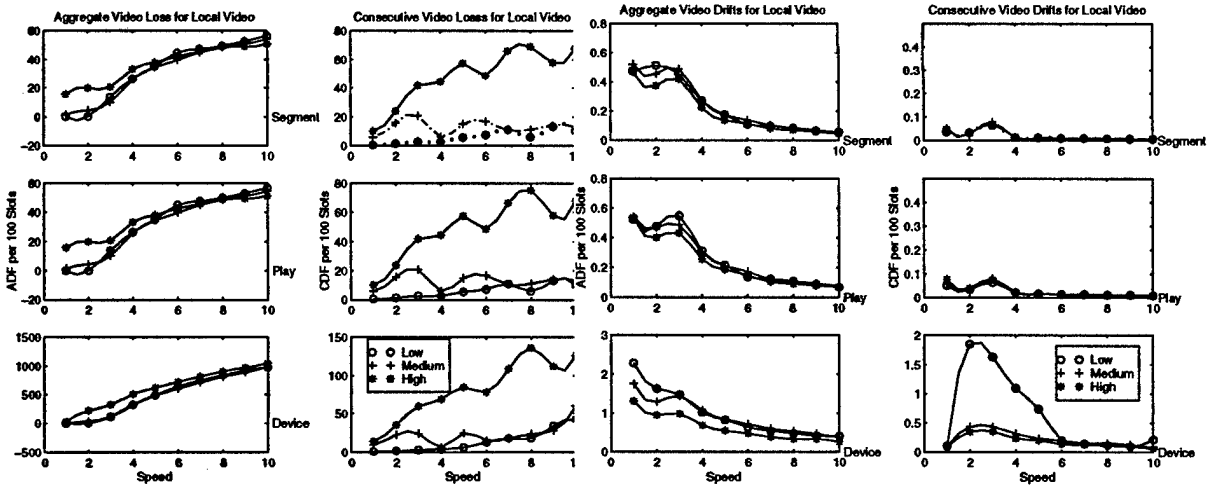

Fig. 3 Video in Local Playback

Processor Loads This section presents results of the local experiment. i.e. the losses of audio and video streams where the `segment` and the `play` object reside on the same machine, and consequently there is no network involved. Our experiments were conducted with differing processor loads at speeds of 30 fps (speed = 1) to 300 fps (speed = 10) for both audio and video. The results are tabulated in terms of our metrics, i.e. aggregate and consecutive losses and drifts, for both media types.

Based on our experimental results we notice the following behavior of CMT.

1. The aggregate and consecutive frame losses increase with respect to speed for all video objects and audio segments.

2. Higher loads affect aggregate losses at low speeds and consecutive losses at higher speeds for all video objects.

3. Processor loads have no effect on audio at the **segment**. We believe that this is due to two conditions: (1) Audio data is relatively small. (2) There is no *delay based drops* for the audio stream at the **segment**, whereas in the video **segment**, drops are based on an *invserse binary order schema.*

4. At higher loads, **play** and **device** objects drop more frames than **segment** object. This is due to the fact, that they are invoked late, thereby drop frames.

5. As the speed increases, drifts drop. That means, as the speed increases, there are fewer frames present, but the ones that aren't dropped appear in time within their slot.

6. At higher loads and low speeds, audio **device** and **play** show larger drifts, indicating jitter.

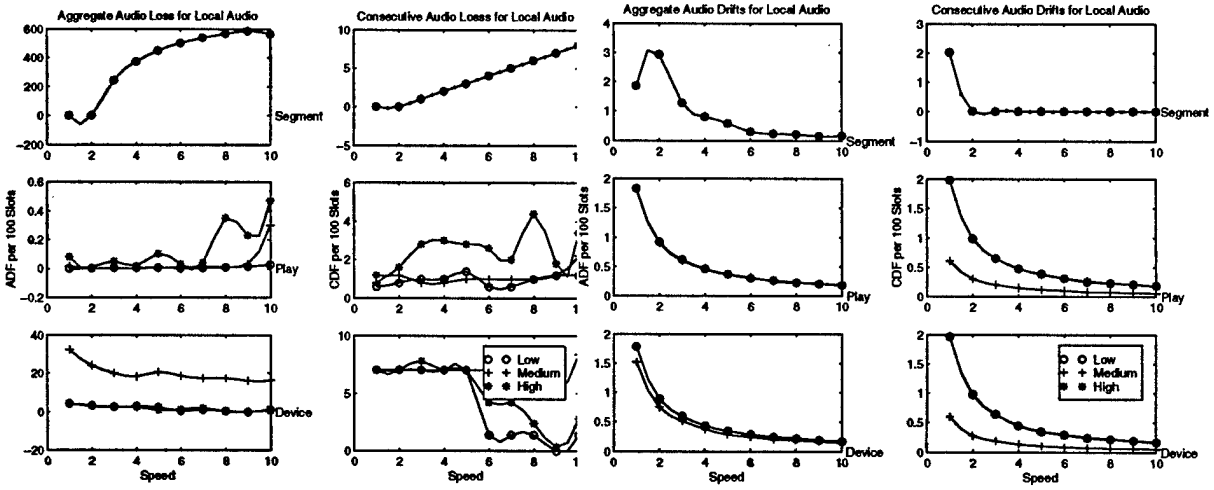

Fig. 4 Audio in Local Playback

Event Loop Loads In this section we present the results of the local experiments for varying event loop loads and varying speeds. The event loop loads vary from 0 to 100 (steps of 20) milliseconds periodically. For these we measure Aggregate and Consecutive Losses and Drifts for both types of media (Audio and Video). The results are summarized as follows:

1. For segment objects retrieving audio, the losses and drifts (both aggregate and consecutive) were constants for varying event loop loads. Higher frame rates (speed) though increased frame losses and drifts. Since the audio data is relatively small compared to video, effect of varying event loop load is minimal at segment object.

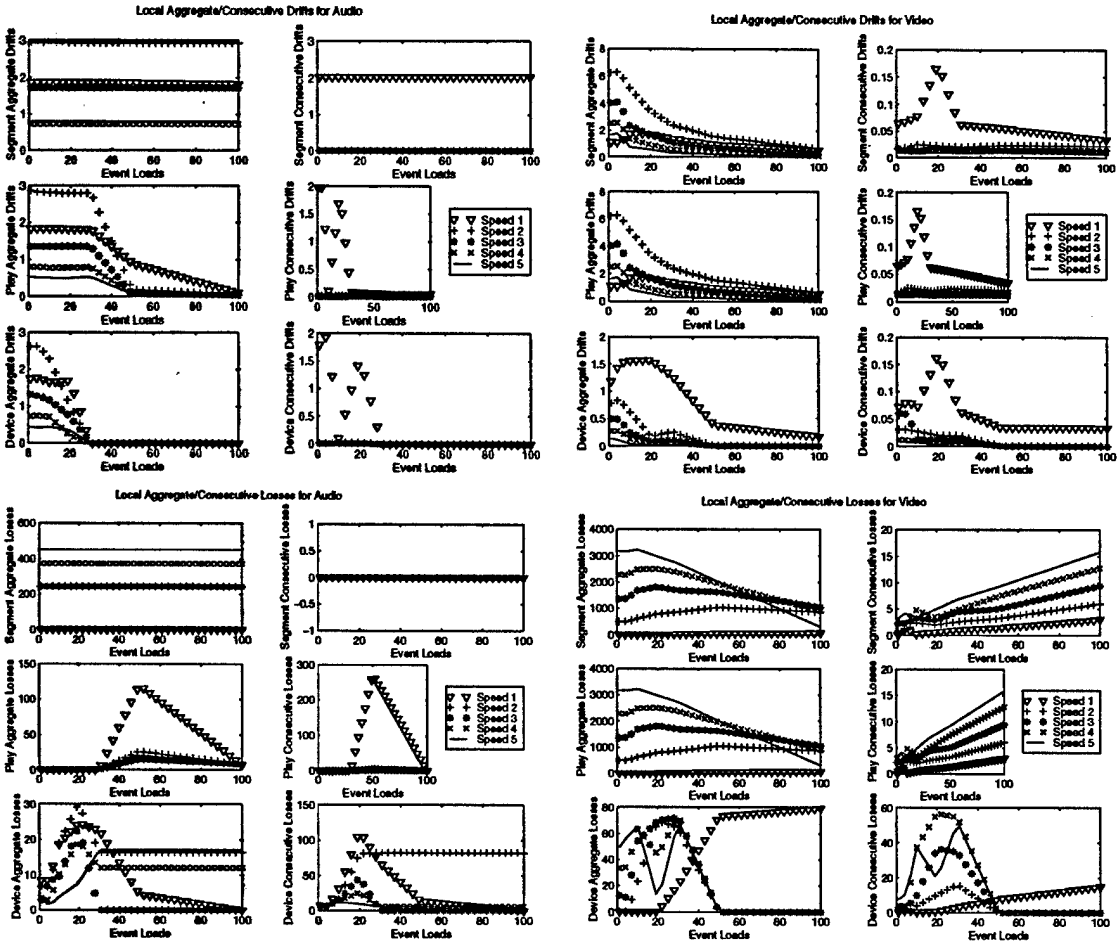

Fig. 5 Losses and Drifts for Audio and Video in Local Playback with Event Loads

2. The only trends visible in play and device objects for audio is the lowering of timing drifts as a result of increased frame losses. Excluding Event loads of up to 40 milliseconds, the variation of audio drifts and losses are minimum i.e. ALF,CLF,ADF,CDF are constants. Thus audio at play and device is more sensitive when event loop loads are in the order of frame rate (30 frames a second, implies a frame in every 33 milliseconds).

3. Another interesting point is that increased speed (frame rate delivery) for audio, has lower losses and drifts (both aggregate and consecutive). This intuitively seems fine as in the same time period, more audio frames are displayed (audio frame size is small) and hence an event loop load which occurs periodically seems to cause lower effect.

4. Video exhibits interesting characteristics. Increased frame rates (speed) show increase in aggregate and consecutive frame losses. What holds true for audio does not hold true here as video frame sizes are much larger than that of audio and the inability of being able to display more frames in a shorter time span more than offsets the lower event loop load effect due to increased speed.

5. Increased event loop loads tends to decrease minimally the aggregate frame losses, while the consecutive losses tend to increase dramatically. This is because of "Bunching Effect" of loss

of frames, which increases consecutive losses, but as a result of deciding to drop a bunch of frames, the overall total number of frames (and hence Aggregate losses) dropped is lower. As can be seen again, lower drifts are achieved (more frames arrive on their time slots) as a result of increased losses.

5.2 Remote Experiment

Processor Loads This section presents results of our networked experiment, i.e. the losses of audio and video streams where the `segment` and the `play` object reside on two different machine connected by a 10 Mbps Ethernet. Accordingly, our experiments were conducted with differing processor loads and network load at speeds of 30 fps (speed = 1) to 300 fps (speed = 10) for both audio and video data.

Processor Loads In this experiment we subjected both the server and the client to varying processor loads, measured in terms of processes waiting for the processor on a 120 MHz Sparc 1 processor. The results are tabulated in terms of our metrics, aggregate and consecutive losses and drifts for both media types.

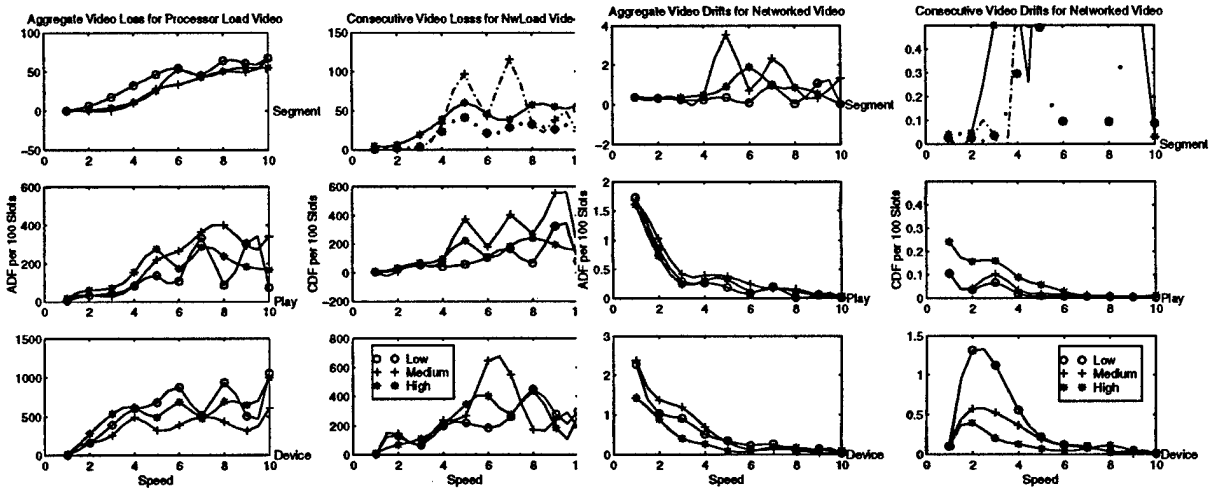

Fig. 6 Video in Networked Playback

Based on our experimental results we notice the following behavior of CMT.

1. In networked video play, as in local play, playout speed increases aggregate and consecutive drops. However the variation of drops appears more wide-spread.
2. In networked audio, the `segment` object is not affected by either the load or speed, and there is wide-spread variation of drops in `play` and `device` objects.

Network Loads In this experiment we subjected the network connecting the server and the client, to varying network loads. (measured in terms of packets per second on a 10Mbps Ethernet) between 0 to 20,000 packets/second, measured by an external *sniffer*. The results are tabulated in terms of our metrics, aggregate and consecutive losses and drifts, for both media types.

Based on our experimental results we notice the following behavior of CMT.

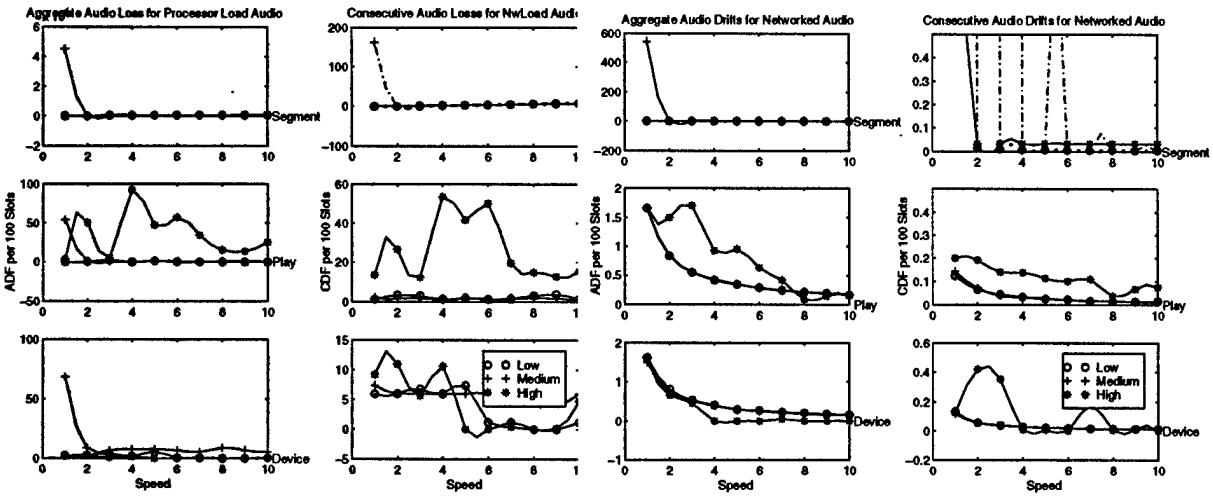

Fig. 7 Audio in Networked Playback

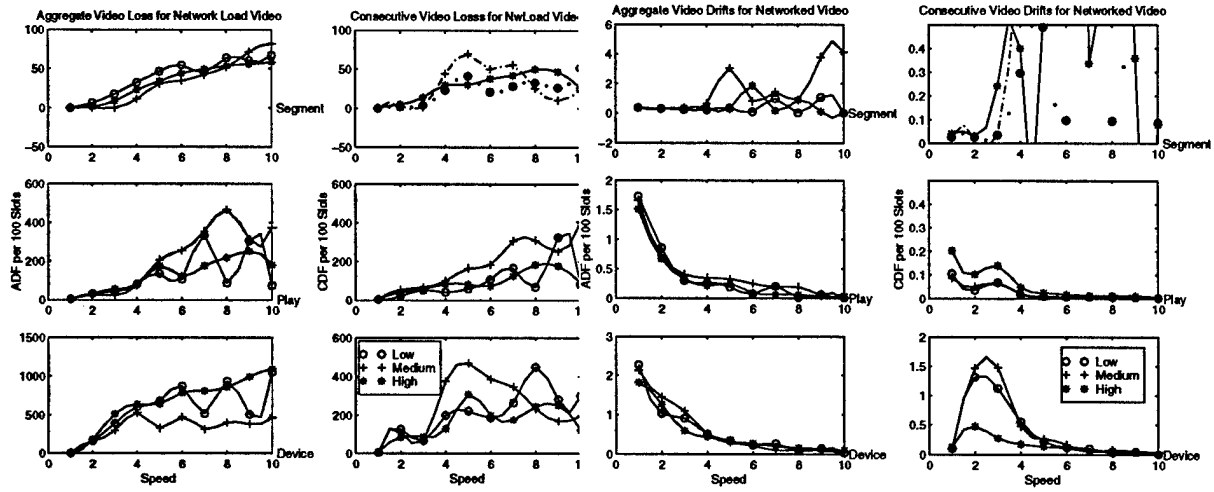

Fig. 8 Video in Remote Playback against Network Loads

1. As in the case of processor loads, we noticed that increasing speed results in dropping frames. Our results show that in the given load ranges, the effect of network loads resulted in wide variations in drop rates.

2. In the audio **segment**, independent of the load range, increased speed result in more audio drops, leading to substantial packet drops at the remote `play` object. This was due to the lateness of audio frames.

3. The drifts decreased monotonically, and independently of network loads, indicating that the frames that were present did appear without much jitter, although as drops indicate many of them were not displayed.

Event Loop Loads In this section we subject both the server and the client to varying event loop loads and varying speeds. The event loop loads vary from 0 to 100 (steps of 20) milliseconds

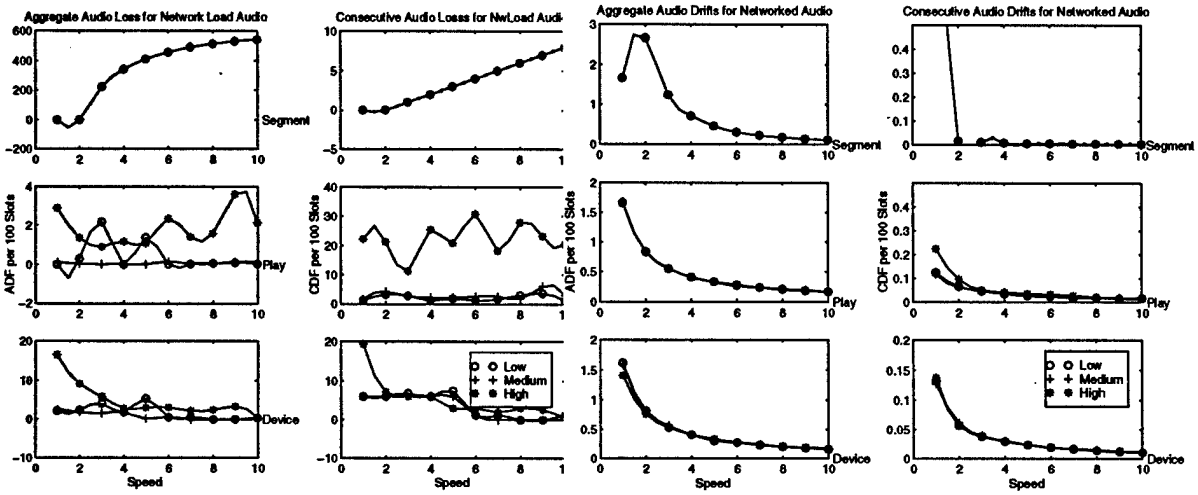

Fig. 9 Audio in Networked Playback against Network Loads

periodically. For these we measure Aggregate and Consecutive Losses and Drifts for both types of media (Audio and Video). The results are summarized as follows:

1. Audio frame losses and drifts exhibit similar patterns like the local playback. Losses and drifts at segment object for audio is virtually none. But the variations progressively magnify from segment to play to device object. This shows that segment objects transmit lot more frames that being displayed at the device, resulting in a lot of wastage of network bandwidth. This emphasizes a need for Client feedback mechanism to be inbuilt in CMT.

2. Video frame losses and drifts again exhibit similar patterns like to corresponding local playback. What is exhibited prominently is the fact that drifts are kept to minimum with progressively increased losses (this occurs with varying event loop loads as well as speed).

6 Synchronization Loss Experiment

6.1 CMT's Performance on Steinmetz' Metrics

This section analyses our data with respect to Steinmetz' parameters. For each type of experiment (i.e. local with differing processor loads, remote with differing processor loads and remote with differing network loads) we show the progress of audio-video drift of a *typical* stream, and the average statistics over five runs. Finally, based on our data we present our conclusions.

The Local Experiment: Processor Loads As shown in Fig 2, segment, play and device objects were used. Their typical behavior with respect to Steinmetz' parameters and the average behaviors of all CM objects are given in Fig 11.

Outcome of the current experiment, as seen from Fig 11, indicates the following facts about CMT's performance:

1. In all CM objects (i.e. segment, play and device), imperceptible audio-video synchronization exists only for about 200 frames. Tolerable synchronization exists for about 400 frames.

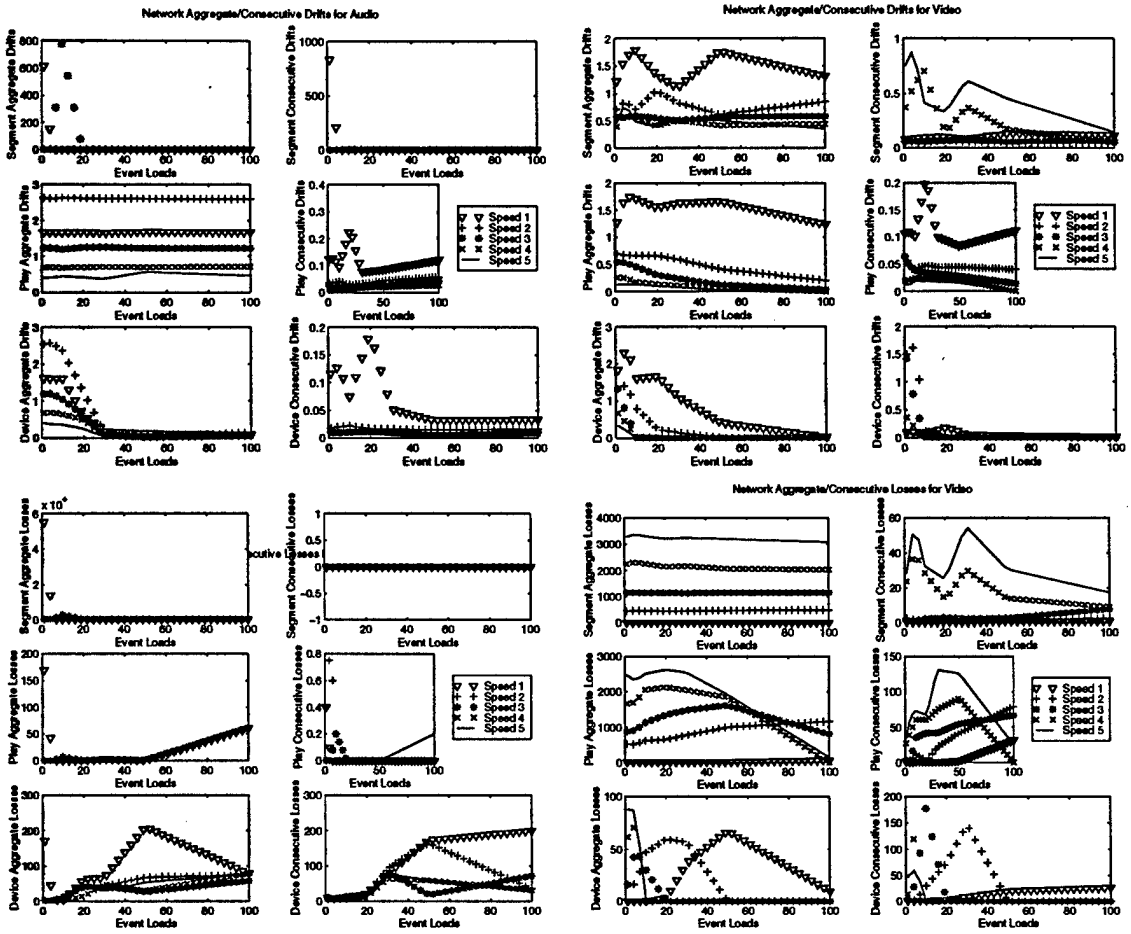

Fig. 10 Losses and Drifts for Audio and Video in Remote Playback with Event Loads

2. Processor load increases audio-video timing drifts.

3. The *segment objects* have a considerable resilience in audio-video timing drift compared to *play* and *device* objects. We believe this behavior is due to the *inverse binary order* [Smi] based drops built into the video segment.

4. In order to provide audio-video synchronization, there needs to be a periodic re-synchronization mechanism in addition to LTS.

5. After an initial increase of audio-video synchronization drift, it appears to show some improvement. Although surprising at first, we soon noticed that under high load both streams drop a considerable number of frames, and thereby show some improvement in synchronization of the frame pairs that are present.

In order to verify our suspicion that the improvement in synchronization mentioned in conclusion 5, i.e. that the slight improvement in synchronization comes at the cost of stream continuity, we measured the frames drops in both streams. These are given in Fig 12.

Based on the graphs for aggregate and consecutive losses in media streams, the following conclusions can be made:

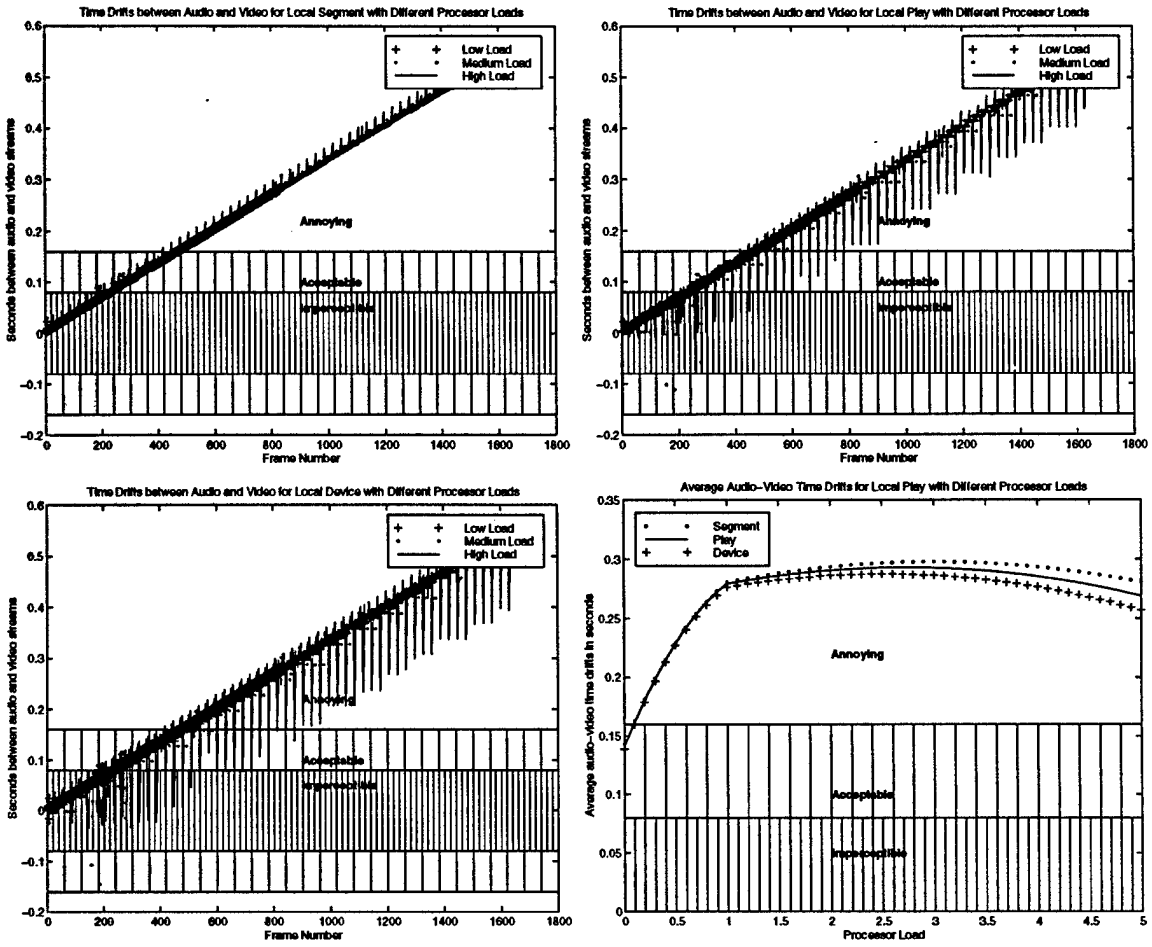

Fig. 11 Synchronization Behavior of Local Playback with Processor Loads

1. The segment does not drop frames within the processor loads that we tested, although there is an in built mechanisms to do so, based on the delay in calling itself. Consequently, the processor loads do not cause such delays, and thus have no effect on the frame dropping mechanism built into the video segment.

2. Although the aggregate drops increase with processor loads, the increment in consecutive drops are minimal in both the play and device objects.

3. The play and device objects drop a significant number of frames, implying that they arrive *too late* to be played. The number of video frames dropped by the device is significantly higher than audio frames.

The Remote Experiment: Processor Loads In this experiment, we subjected the simple remote playback to varying processor loads. These processor loads were generated by a separate process on the same CPU, and applied to both the client and the server. As described in Section 6.1, the experiment involved segment, play, device, and the network twins packet source and packet destination. As in previous experiments, typical behavior of segment, play and device and their average behavior measured with respect to Steinmetz' metrics are given in Fig 13.

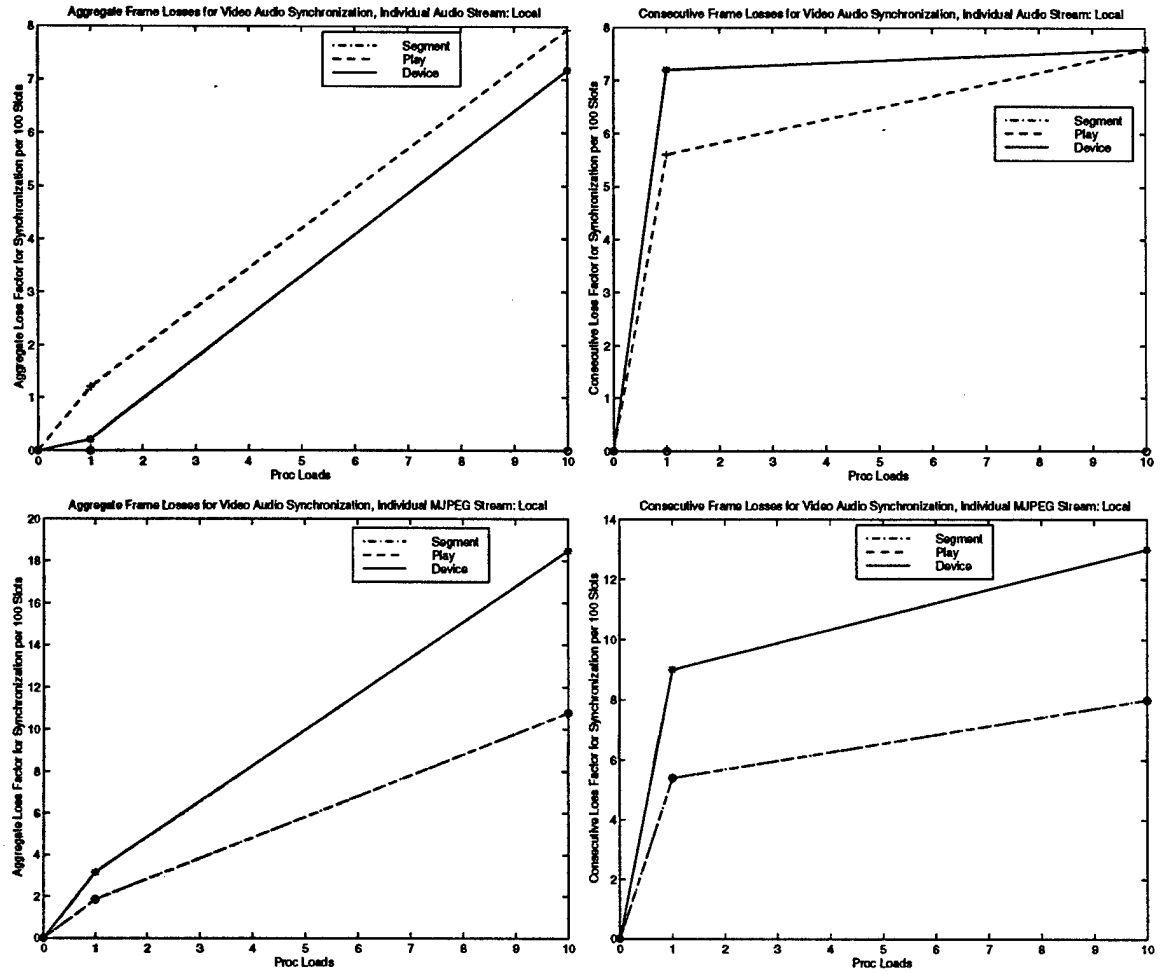

Fig. 12 Frame Losses in Local Playback with Processor Loads

Outcome of the current experiment, as seen from Fig 13, indicates the following facts about CMT's performance over a local area network:

1. None of the CM objects have tolerable audio-video synchronization for any appreciable amount of time.
2. The segment object is not affected by the processor load as much as others.
3. There is vastly varying mis-synchronization between remote play and remote device objects during high processor loads.
4. The remote device and play objects appears to be gaining audio-video synchronization among the very few frames displayed on the average.

Based on the graphs for aggregate and consecutive losses in media streams, the following conclusions can be made:

1. The audio segment drops a constant amount of frames when there are processor loads both locally and remotely.

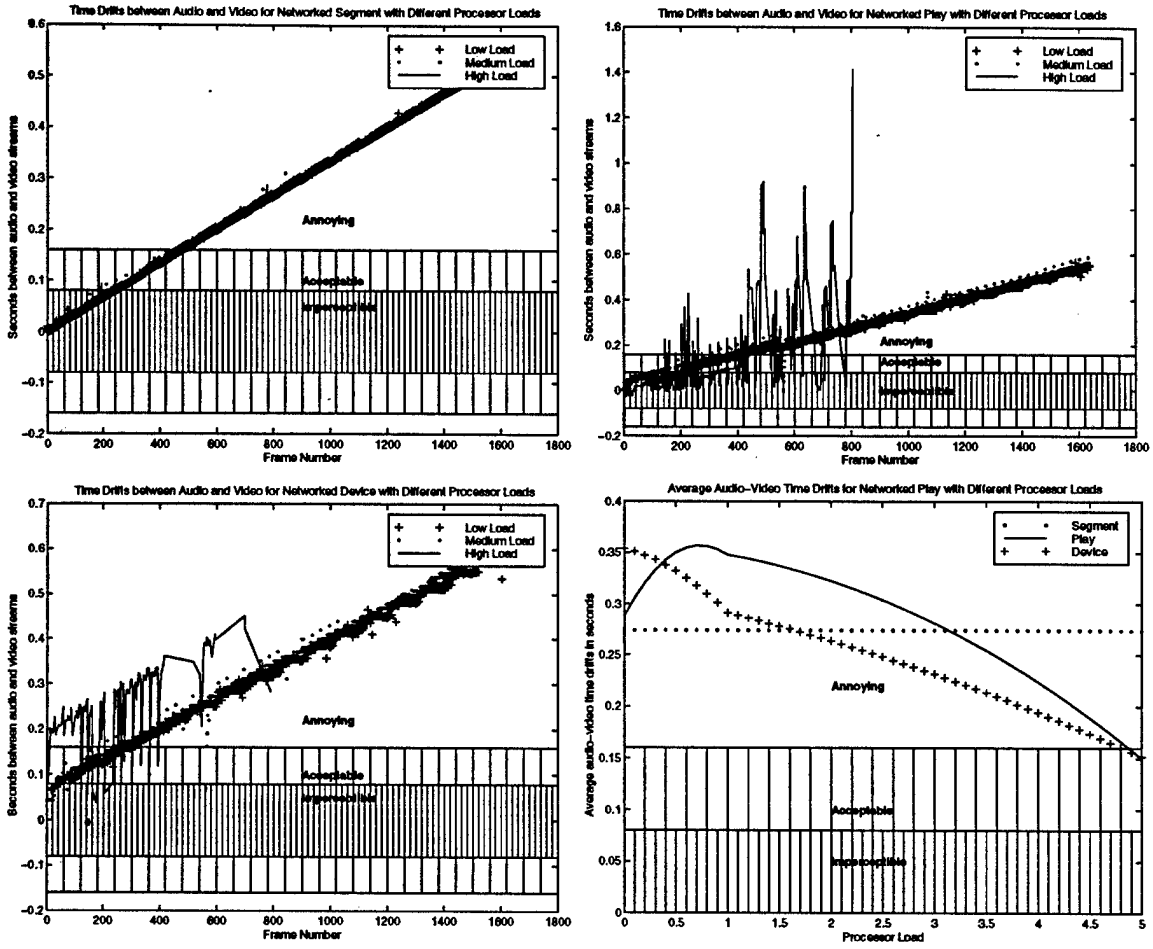

Fig. 13 Synchronization Behavior of Remote Playback with Processor Loads

2. The frame drop procedure built into the video segment works remarkably well in the sense that the increase in video frame drops are kept to a minimum at the segment.

3. Due to the network, a significant number of frames, both audio and video are lost at both the device and play objects sitting at a remote client.

4. Consecutive loss in frame drops increases as the processor loads increase in remote play.

The Remote Experiment: Network Loads In this experiment, we subjected the simple remote play to varying network loads. As described in Section 6.1, the experiment involved segment, play, device, and the network twins packet source and packet destination. As in previous experiments, typical behavior of segment, play and device and their average behavior measured with respect to Steinmetz' metric are given in Fig 15.

Outcome of the current experiment, as seen from Fig 15, indicates the following facts about CMT's performance over a local area network:

1. Remote audio-video playback is not within humanly tolerable synchronization limits.

2. As the stream progresses, the audio-video mis synchronization increases without bound.

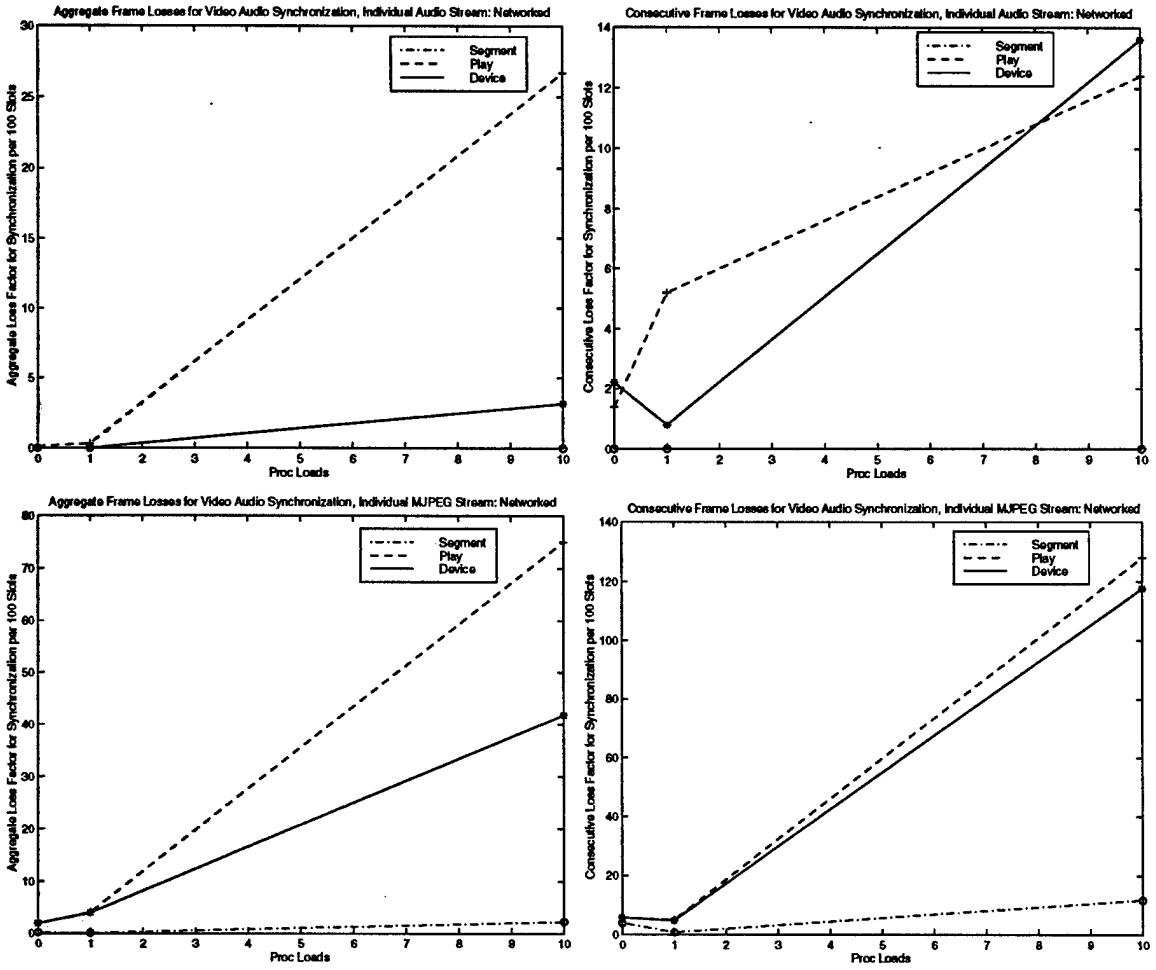

Fig. 14 Frame Losses in Remote Playback with Network Loads

3. Segment objects are not affected by network loads. This is natural as they are at the server site, and do not depend on the network at all.

4. The variation of audio-video synchronization critically affects the play object, and consequently the device object. The degree of mis synchronization is much higher than the affect of high processor load. The actual number of frames dropped are given in Fig 16.

5. After some time, higher network loads result in loosing large amount of frames, so that the remaining frame pairs tend towards acceptable synchronization.

The Local and Remote Experiment: Event Loads In these set of experiments, we measured the effect of event loop loads on the synchronization behavior of CM streams. The behavior of the streams were similar in both the local as well as remote play experiments. We are including only the graphs for remote play to show the synchronization characteristics of the streams. Also we have included event loop loads of 1, 10, 30 and 100. Event loop loads of 5 and 20 had same pattern as 1 and event loop load of 50 had same pattern as 30. Hence we did not include them in figure 17.

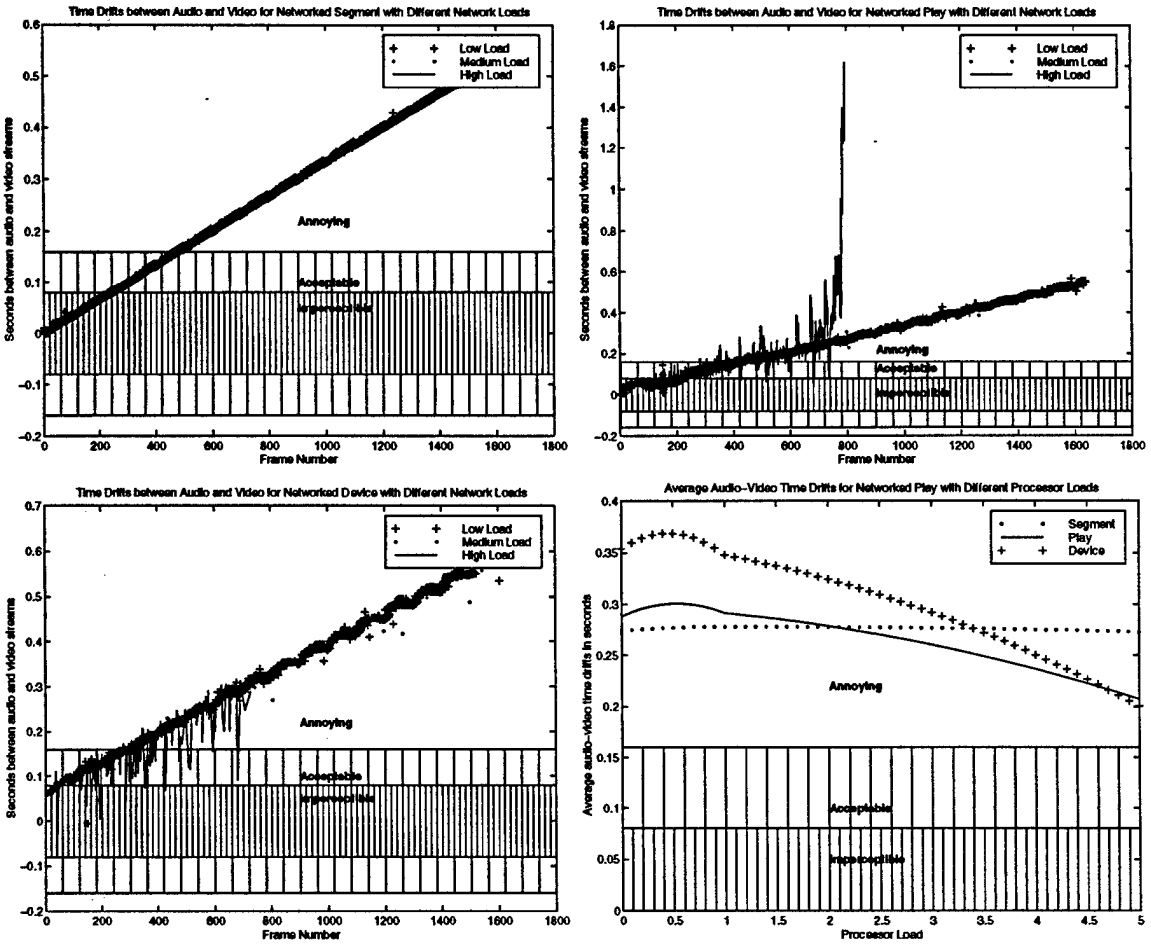

Fig. 15 Synchronization Behavior of Remote Playback with Network Loads

1. Tolerable audio-video synchronization is achieved to about 1500 frames. Then it exponentially degrades. This pattern is seen in all objects (segment, play and device).
2. Increased event loop loads increase the synchronization drifts (of much lower quality than the tolerable limits specified by steinmetz), especially after 1500 frames.
3. Synchronization drifts at device is higher than those at other objects. This clearly shows that there is a need for segment and other objects at the server to be aware of the state of synchronization. Again a need for feedback.

Summarizing, As the stream progresses, the degree of asynchrony increases without bound. Remote Synchronization quality is worse than local, those the patterns shown in each of them were similar. Based on frame drop graphs of Fig 16 and Fig 14, we notice that the effects of network load and processor load play a similar role in frame drops and maintaining synchronization.

Conclusions Based on our experiments, when the client and the server objects are at the same site, audio and video are synchronized to be within tolerable limit in terms of Steinmetz' metrics only within the first 400 or so frames; i.e. first 13 seconds of playback. In the case of remote play, there is hardly any synchronization at all at the client site.

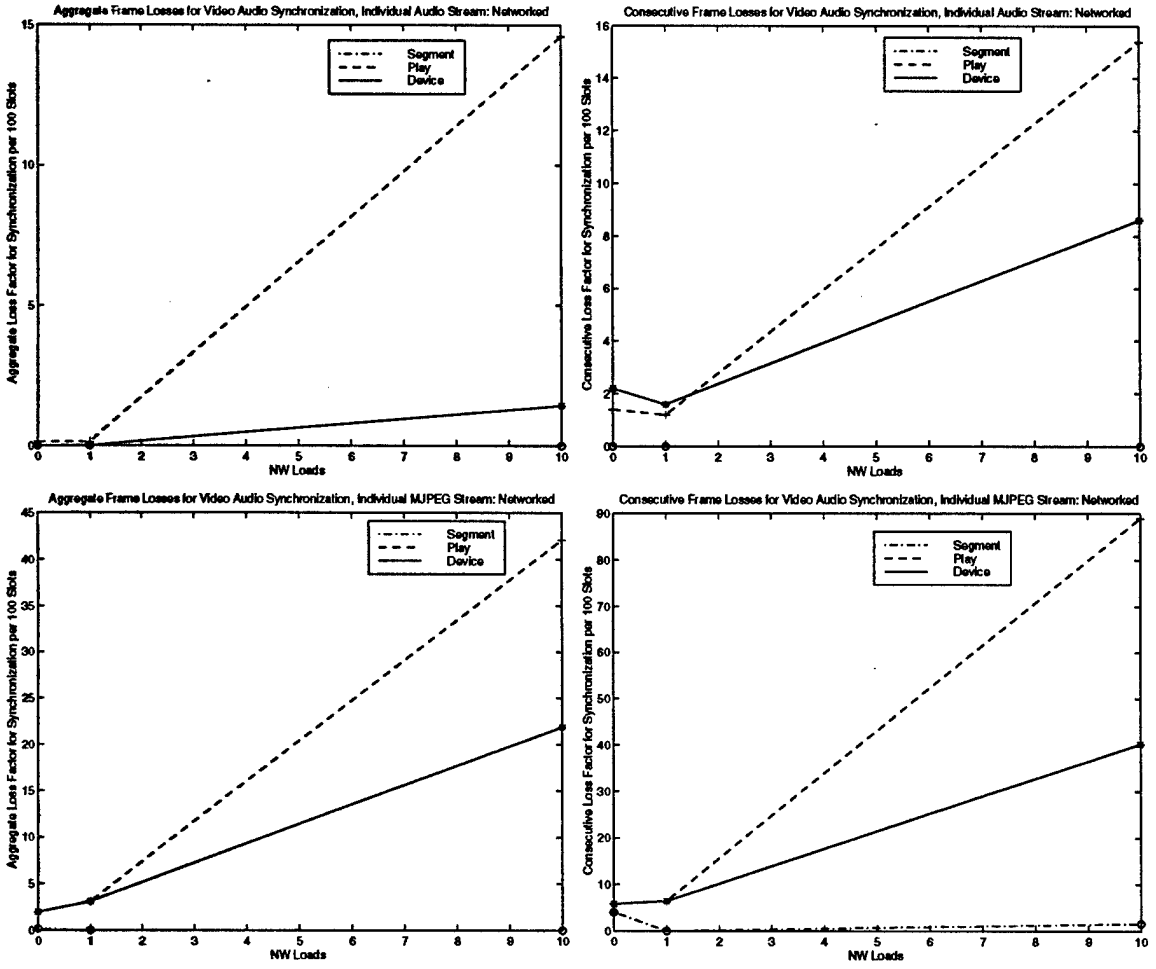

Fig. 16 Frame Losses in Remote Playback with Network Loads

In order to maintain synchronization, for local clients and servers, some solution that incorporates periodic synchronization, such as [RR93, Sri92] may be required.

One of the major problems we encountered in applying Steinmetz' metrics to audio-video synchronization was the fact that the metrics were designed to be applied to synchronization between loss-less media streams, and those provided by CMT are lossy services. Consequently, we measured our traces with respect to lossy synchronization metrics described in Section 2.

6.2 CMT's Performance on Lossy (Synchronization) CM Metrics

In this section, we analyze our data against synchronization metrics for lossy continuous media presented in Section 2.

In these metrics, mis-synchronizations between audio and video streams were measured in terms of aggregate and consecutive losses of content, and aggregate and consecutive drifts. As described in Section 2, at the heart of these metrics lies the concept of *slots* to which media frames are suppose to belong. The difference in frame numbers belonging to a given slot measures a *unit content loss*, and aggregate and consecutive non zero such losses provide metrics of lossy synchronization. Time drifts are the aggregate and consecutive drifts within each such slot. Consequently, the so called

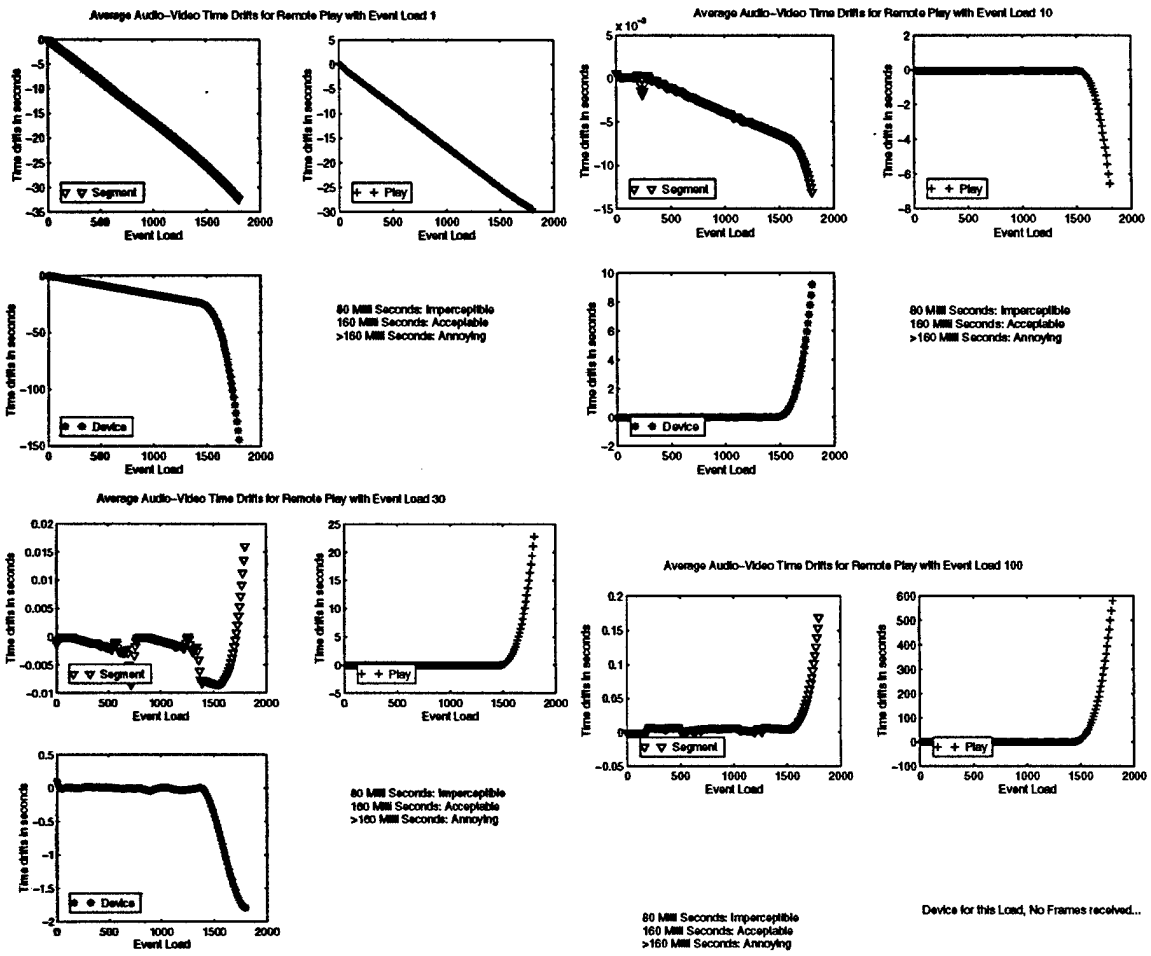

Fig. 17 Synchronization Behavior of Remote Playback on Event Loads

content losses provide a coarse measure and drift parameters provide a finer measure. The values obtained for these metrics in our experiments are given in the following sections.

The Local Experiment: Processor Loads The graphs in Fig. 18, viewed in light of the loss graphs of Fig. 12 and their comparison with those given for Steinmetz' metrics in Fig. 11, indicate the following:

1. From the very beginning there are aggregate synchronization losses between audio and video streams, although with processor loads they do increase. The difference here is more pronounced than in Steinmetz' metric. As the loads increase, the individual streams drop more frames, and consequently at the cost of loss of continuity, there is a marginal decrement in synchronization losses.

2. The device object always has a lower asynchrony, as it is passed only those frames that have arrived in time. This is evidenced in the higher aggregate losses for audio and video streams in Fig. 12, and in the aggregate synchronization graphs for devices of both streams given in Fig. 18.

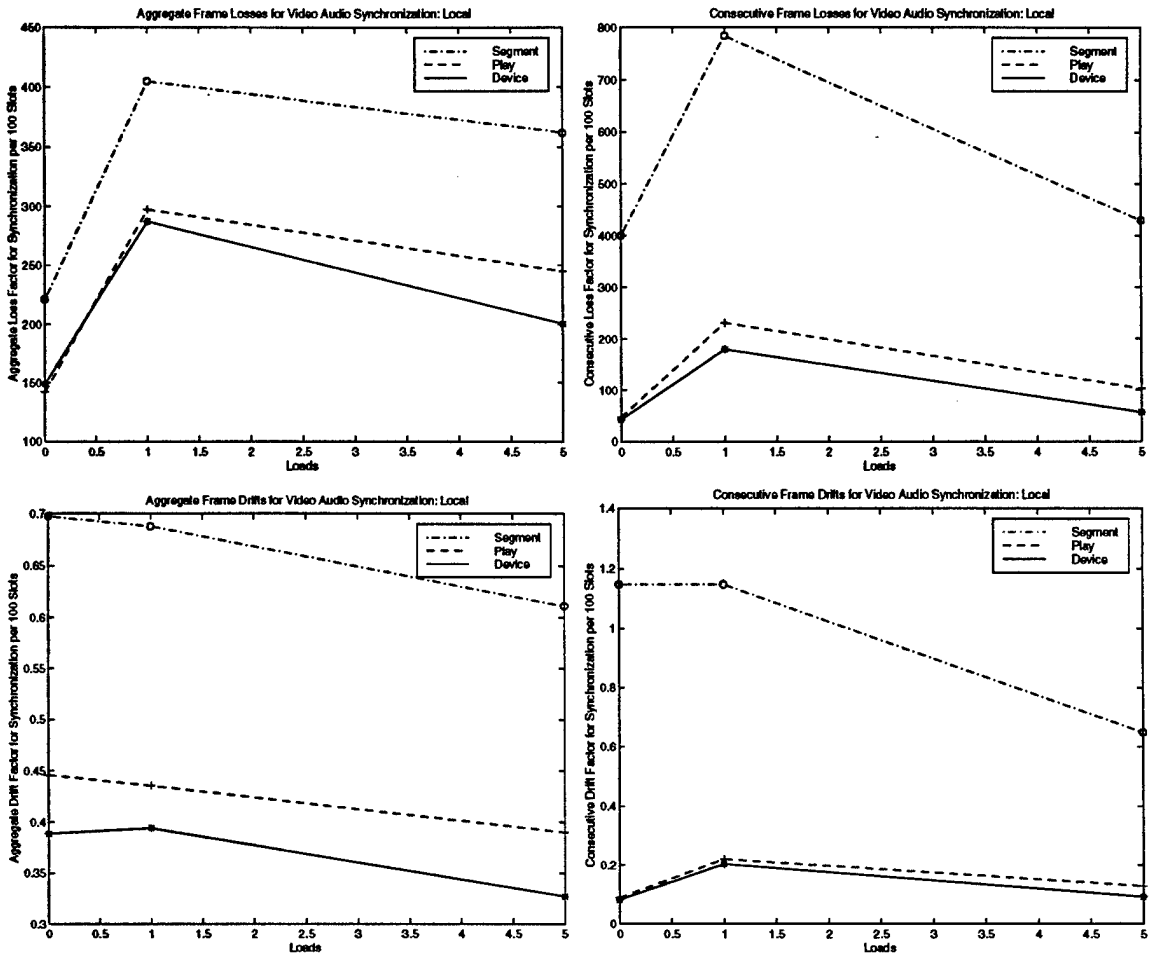

Fig. 18 Synchronization Behavior of Local Playback

The Remote Experiment: Processor Loads From the results of Fig. 19, we can draw the following conclusions.

1. As the processor loads increase, the loss of synchronization at segments, play objects and devices increase, and the increments are more than the corresponding increments in the local play.
2. From Fig. 14, we notice that aggregate frame losses remained constant over processor loads. Nevertheless, according to Fig 13, synchronization losses remained high, but constant. But according to Fig 19, the aggregate synchronization loss factor (ASLF) in Section 2 increases. The reason is that because the drops in both segments are un-coordinated, and only the pairs that are shown have approximately the same time gaps, Steinmetz' metric does show a constant value. But ALSF shows the loss of synchrony caused by un-coordinated drops between the two streams.

The Remote Experiment: Network Loads Graphs of Fig. 20 indicate the following about the behavior of remote playback in the presence of network load.

1. The aggregate synchronization losses in the segment remains constant. Also, from Fig. 14 audio was not dropped, and video was dropped at a constant rate. Therefore, the loss between the

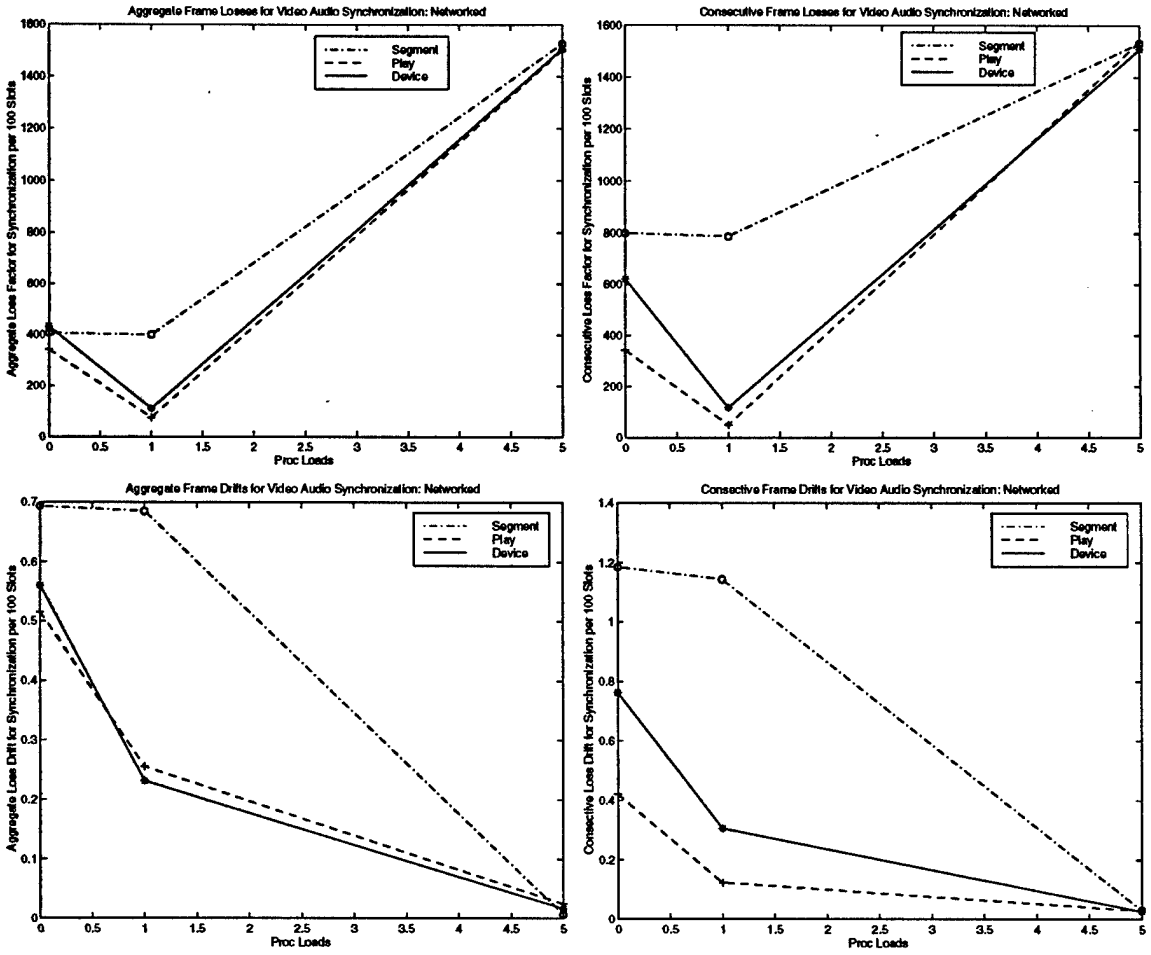

Fig. 19 Synchronization Behavior of Remote Playback with Processor Loads

two streams were *un-intentionally coordinated*, resulting in a constant ASLF over network loads. This is expected since the network should not affect the server, rather only the client (if at all). Rightfully, according to Fig. 15 Steinmetz' metric also indicates this trend.

2. As the graph indicates, the client side is badly affected by the network load, and both at the device and the play objects there is a remarkable drop in ASLF as the network load increases, which is somewhat surprising. Nevertheless, these values are far beyond the acceptable parameter ranges reported in the human perception study of [WSNF99].

The Local and Remote Experiment: Event Loads In these set of experiments, we measured the effect of event loop loads on the individual (each stream) frame losses and drifts of CM streams. The behavior of the streams were similar in both the local as well as remote play experiments. We are including only the graphs for remote play to illustrate the behavior. Also the effects on audio and video streams were similar except the variations in audio were smaller. Hence we included the individual drifts for video (did not include audio) and individual losses for audio (did not include for video) to show what the individual stream patterns are in figure 21.

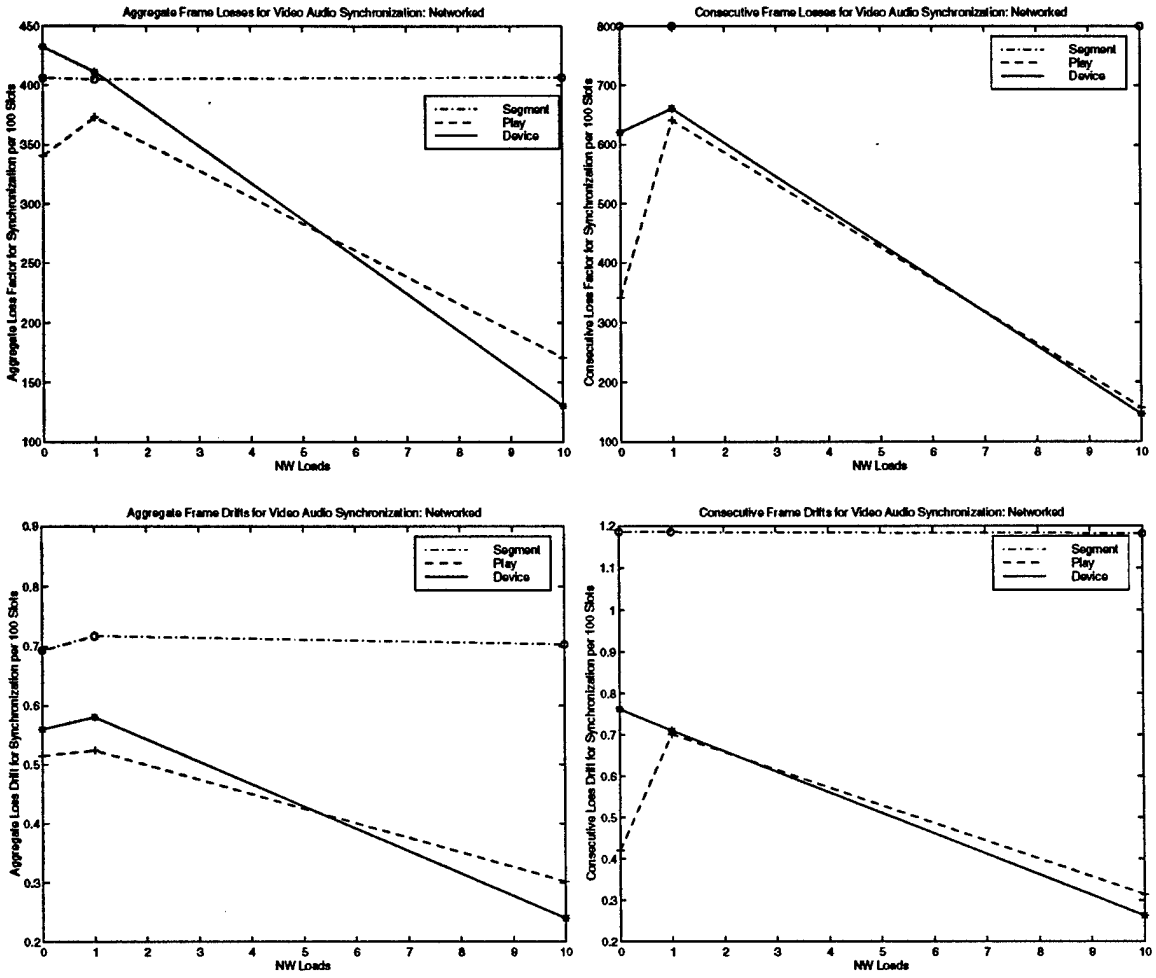

Fig. 20 Synchronization Behavior of Remote Playback with Network Loads

1. For video, individual frame drifts decrease as a result of exponentially increasing frame drops. This pattern is again seen in audio, but with a smaller variation.

2. As what was seen in steinmetz metrics, those frames that did arrive in time, the synchronization drifts were small. But this is solely attributed to fact that there were very few of such frames. This pattern is clearly reflected in the graph for synchronization losses which were very high and showed that synchronization is not within human tolerable limits.

The above pattern was seen in both local and remote, except that the synchronization on remote playback was worse than local playback.

Conclusions Even by the lossy metrics of [WS96], as in the case of Steinmetz' metrics, synchronized play of audio-video streams in CMT quickly reaches outside the bounds of human tolerance. Consequently, as stated in Section 6.1, there is a need for synchronization services on top of CMT.

One of the problems we encountered in subjecting the data from our CMT experiments to the loss media metrics is the fact that when the *mjpegSegment* intentionally drops packets, it also interpolates the beginning and end times for video frames. This has two effects on our experiments.

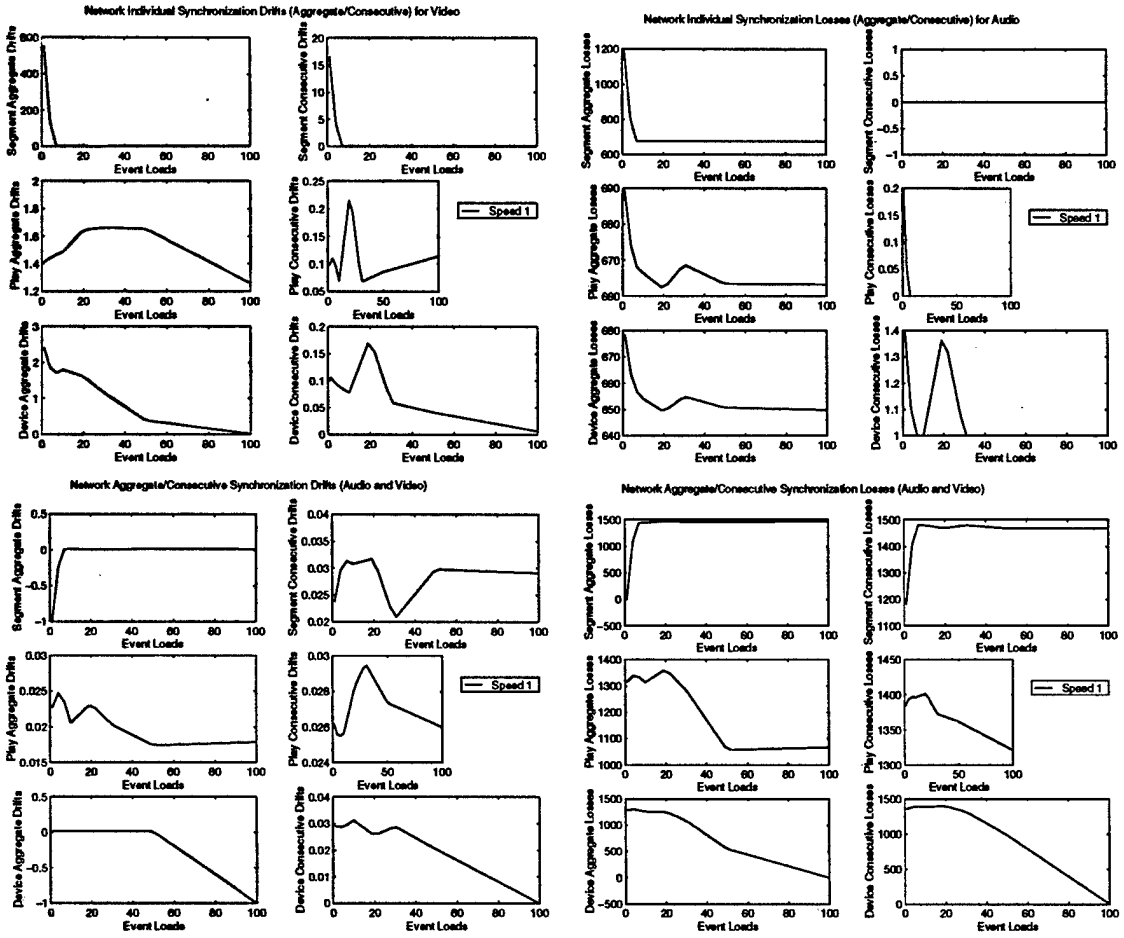

Fig. 21 Frame Losses in Remote Playback with Event Loads

Firstly, it violates the uniform time length of slots as the metric assumes, thereby casting doubts of the applicability of these metrics to measure CMT's services. Conversely, we are unaware of any metric that can be applied to measure lossy synchronization where the slotting is allowed to be varied by the service provider at will. Similarly, we have mentioned the problems encountered in applying Steinmetz' metrics, as they are not designed to measure lossy synchronization.

Secondly, and more importantly, it destroys the ability to precisely synchronize downstream, as the new time stamps are not exactly what can be used to synchronize.

Reiterating, the reasons for such poor synchronization can be attributed to the following causes:

1. Both streams loose synchronization at the **segments** because they only rely on a common timeline, LTS. As shown in [Ste96] this type of control is insufficient for imperceptible quality of *lip synchronization*. Thus there is a need for better mechanisms.

2. Both streams, once started, have no other points of re-synchronization. The need for this is apparent, as for about 400 frames, the two streams retain imperceptible mis-synchronization, and for about further 150 there is tolerable mis-synchronization.

3. Due to the independence of the component streams, they have two independent fetch cycles, where the two streams are woken up by two separate events, and schedule themselves to be woken

up after the specified period. In case of heavy processor or event loop loads, two corresponding wake-up events could be far apart in time.

Due to stated reasons, we notice that the root cause of intolerable lip synchronization is due to the independence of the component streams. Consequently, as a solution, we propose *Stream Groups*, an abstraction with accompanying design and implementations for improving lip synchronization in CMT. The details of this work is not listed in this paper, and are worked out in [PVW+, Par97]. The stream groups abstraction is proposed to capture the requirements of fine grain synchronization between lossy CM streams. The application should be able to specify the following for synchronized display of CM streams:

- File names, sources and sinks for component streams.
- QoS parameters for component streams. For audio and video, in our current setup, these are aggregate and consecutive media losses and timing drift, rates and their variations, and start and end points in a given file containing the stream.
- QoS parameters for the group of synchronized streams. In the current stage of development, these are the QoS parameters for synchronization: i.e. aggregate and consecutive synchronization losses and timing drifts.

7 Conclusions and Future Work

In a series of experiments, we have investigated the synchronization services provided by CMT. While it is the best continuous media programming testbed we are aware of, to provide synchronized audio-video services that are to be within humanly tolerable limits, there need to be additional mechanisms.

In our experiments we subjected CMT's audio-video services to metrics designed and tested for human tolerances of mis-synchronizations by Steinmetz'. We found that for a single site play there is imperceptible mis-synchronization only for about 10 seconds, and tolerable display only for about three seconds more. Furthermore, for a client and server connected by a relatively uncongested Ethernet, there seems to be no tolerable synchronization for any appreciable time limit at all. Because CMT drops media streams, and Steinmetz' metrics assumed that media streams were loss-less, we analyzed the same results with respect to another set of metrics and arrived at the same result. We have also encountered problems with applying these metrics to CMT data as the built-in drop policy for video alters the time stamps in video streams. In addition, this policy is likely to interfere with any time based re-synchronization that may be attempted by other objects downstream.

We believe that synchronization problems in CMT arise due to the lack of coordination between the two objects providing the same service for the two supposedly synchronized streams. At the implementation level they are woken up by two different events, and follow their own cycles for CM services. Also, in the network transmission there is no effort to coordinate the two streams together. We are currently working on providing a synchronization service for CMT as a partial solution to these problems [PVW+]. We have provided some bursty error spreading techniques to improve perceptual quality of CM stream when transmitted over a lossy network in [NVS99].

As stated, in applying Steinmetz' metrics to CMT data we encountered the problem of frame drops; and independently we have measured them under differing load conditions [WVP+98]. In future, we plan to make CMT drop frames based on user specified QoS thresholds.

We have presented results of a performance evaluation of media losses in audio and video streams in CMT with respect to a set of metrics designed to measure continuity losses and timing drifts. The general trend of our results indicate that, as expected, at higher speeds, CMT looses more frames. In addition, loads increase aggregate losses at lower speeds and consecutive losses at higher speeds. It was also observed that at higher speeds, at the cost of loosing more frames, the ones that are retained have smaller timing drifts.

The primary objective for this experiment was to motivate the need for QoS provisioning in CM streams and synchronization between groups of such CM streams. CMT is an ideal environment to experiment with and implement such mechanisms. Towards achieving this goal, the effect of media losses in CMT on audio-video synchronization has also been investigated in 6.

Our future work include making frame drops in CMT based on user specified QoS parameters. Towards this end, we are replacing appropriate object in CMT with corresponding objects that have QoS based drop policies encoded in them. Furthermore, in order to provide proper QoS provisioning at play objects, it is necessary for clients (`play` objects) to have a feed back mechanism toward the server (`segment`) objects.

References

[AFKN94] R.T. Aptekar, J.A. Fisher, V. Kisimov, and H. Nieshlos. Distributed Multimedia:User Perception and Dynamic QoS. In *SPIE* , volume 2188. SPIE, 1994.

[AFKN95] R.T. Aptekar, J.A. Fisher, V. Kisimov, and H. Nieshlos. Video Acceptability and Frame Rate. *IEEE Multimedia*, pages 32–40, September 1995.

[MPR97] Ketan Mayer-Patel and Lawrence A. Rowe. Design and Performance of the Berkeley Continuous Media Toolkit. In Martin Freeman, Paul Jardetzky, and Harrick Vin, editors, *Proceedings of Multimedia Computing and Networking*, pages 194–206, 1997.

[NVS99] H. Ngo, S. Varadarajan, and J. Srivastava. Error Spreading: Reducing Bursty Errors in Continuous Media Streaming . *Submitted to IEEE, ICMCS '99*, 1999.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, Massachusetts, 1 edition, 1994.

[Par97] Shwetal S. Parikh. Stream groups: Synchronization services for the Continuous Media Toolkit. Master's thesis, University of Minnesota, December 1997.

[PVW+] Shwetal Parikh, Srivatsan Varadarajan, Duminda Wijesekera, Jaideep Srivastava, and Anil Nerode. Stream Groups: A QoS based Synchronization Support Service for the Continuous Media Toolkit. Working Paper at the University of Minnesota.

[RR93] S. Ramanathan and P. Venkat Rangan. Feedback Techniques for Intra-Media Continuaity and Inter-Media Synchronization in Distributed Multimedia Systems. *The Computer Journal*, 36(1):19 – 33, 1993.

[SB96] Ralf Steinmetz and Gerold Blakowski. A media synchronization survey: Reference model, specification and case studies. *IEEE Journal on Selected Areas in Communication*, 14(1):5–35, 1996.

[SE93] R. Steinmetz and Clemens Engler. Human perception of media synchronization. Technical Report 43.9310, IBM European Networking Center, Heidelberg, 1993.

[Smi] Brian Smith. Cyclic-UDP: A Priority Driven Best-Effort Protocol. Paper available from the web at http://www.cs.cornell.edu/Info/Faculty/Brian_Smith.html/Publications.

[SNL95] Brian K. Schmidt, Duane Northcutt, and Monica Lam. A method and apparatus for measuring media synchronization. In T.D.C. Little, editor, *NOSDAV 95*, pages 190–201, September 1995.

[Sri92] Cormac John Srinan. *Synchronization Services for Digital Continuous Media.* PhD thesis, University of Cambridge, October 1992.

[SRS93] Brian Smith, Lawrence A. Rowe, and Yen S. Tcl distributed programming. In *Proceedings of the 1993 Tcl/Tk Workshop*, 1993.

[SRY93] Brian Smith, Larry Rowe, and S Yen. A Tcl/Tk Continuous Media Player. In *Proceedings Tcl 1993 Workshop*, June 1993.

[Ste96] Ralf Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communication*, 14(1):61–72, January 1996.

[Wel95] Brent B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall, New Jersey, 1 edition, 1995.

[WS96] Duminda Wijesekera and Jaideep Srivastava. Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications*, 3(1):127–166, September 1996.

[WSNF99] Duminda Wijesekera, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Experimental Evaluation of Loss Perception in Continuous Media. *to appear in ACM - Springer Verlag Multimedia Journal*, July 1999.

[WVP+98] Duminda Wijesekera, Srivatsan Varadarajan, Shwetal Parikh, Jaideep Srivastava, and Anil Nerode. Performance evaluation of media losses in the continuous media toolkit. In *International Workshop on Multimedia Software Engineering , MSE'98, Kyoto, Japan*, 1998.

This article was processed using the LaTeX macro package with LLNCS style